7-2017

Mining diverse consumer preferences for bundling and recommendation

Ha Loc DO
Singapore Management University, hldo.2012@phdis.smu.edu.sg

MINING DIVERSE CONSUMER PREFERENCES FOR BUNDLING AND RECOMMENDATION

DO HA LOC

SINGAPORE MANAGEMENT UNIVERSITY
2017

MINING DIVERSE CONSUMER PREFERENCES for BUNDLING and RECOMMENDATION

by

Do Ha Loc

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Hady W. Lauw
Assistant Professor
Singapore Management University

David Lo
Associate Professor
Singapore Management University

Jiang Jing
Associate Professor
Singapore Management University

Gao Cong
Associate Professor
Nanyang Technological University

Singapore Management University
2017

MINING DIVERSE CONSUMER PREFERENCES

for BUNDLING and RECOMMENDATION

by

Do Ha Loc

Abstract

That consumers share similar tastes on some products does not guarantee their agreement on other products. Therefore, both similarity and difference should be taken into account for a more rounded view on consumer preferences. This manuscript focuses on mining this diversity of consumer preferences from two perspectives, namely 1) between consumers and 2) between products.

Diversity of preferences between consumers is studied in the context of recommendation systems. In some preference models, measuring similarities in preferences between two consumers plays the key role. These approaches assume two consumers would share certain degree of similarity on any products, ignoring the fact that the similarity may vary across products. We take one step further by measuring different degrees of similarity between two consumers. Specifically, we propose a probabilistic framework *CAM-DPMF* to capture the extent two consumers agree in their preferences on a particular product based on the corresponding product ratings.

Diversity of preferences between products is reflected in observations that any two products would attract different consumer groups. The intuition is the basis for product bundling. There are two types of product bundling. Pure bundling is the a bundling form in which either a bundle or its components are available to the market. If both the bundle and its components are offered, we call it mixed bundling. Since it is impractical to offer all possible bundles to market, firms are interested in a selective set of bundles satisfying some objectives and constraints. Finding such optimal bundle set is computationally

challenging due to diversity of demand for each bundle. In this dissertation, we present computationally efficient approaches to look for profit-maximizing bundle sets in two forms, namely bundling configuration and top-K bundles.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I am indebted to my advisor, Assistant Professor Hady W. Lauw, for his continuous support, guidance, and encouragement, and for providing me with the best working environment throughout my PhD study. The completion of this dissertation is made possible partly by his assistance and efforts given to me.

I do appreciate valuable comments and suggestions from members of my dissertation committee: Associate Professor David Lo, Associate Professor Jing Jiang, and Associate Professor Gao Cong.

It is my fortunate to have supports from my fellow researchers, seniors and friends in Living Analytics Research Centre (LARC): Philips K. Prasetyo, Agus T. Kwee, Ibrahim Nelman Lubis, Arinto Murdopo, Freddy C. T. Chua, Hoang Tuan Anh, Luu Minh Duc, Pei Hua and Desmond Koh.

I would like to commend on the excellent administrative assistance given to me by the staffs in School of Information Systems: Ong Chew Hong and Seow Pei Huan; and the staffs in LARC: Chua Kian Peng, Cheong Su Fen, Angela Kwek Renfeng, Sheryl Chen, Phoebe Yeo, Alenzia Wong Poh Luan, Fong Soon Keat, Desmond Yap, Janice Tan, and Nancy Beatty.

Last but not least, I would like to thank Singapore Management University for the scholarships and financial supports for my study. My research work is funded by The Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

Dedicated to my family,

For their endless love, support and encouragment.

List of Publications

The work discussed in this dissertation has resulted in the following international publications. We list these publications of the author during his Ph.D. candidature below.

(a) Do, Loc, and Hady W. Lauw. "Probabilistic Models for Contextual Agreement in Preferences." ACM Transactions on Information Systems (TOIS) 34.4 (2016): 21.

(b) Do, Loc, Hady W. Lauw, and Ke Wang. "Mining revenue-maximizing bundling configuration." Proceedings of the VLDB Endowment 8.5 (2015): 593-604.

(c) Do, Loc, and Hady W. Lauw. "Modeling contextual agreement in preferences." Proceedings of the 23rd international conference on World wide web. ACM, 2014.

Chapter 1

Introduction

Consumer Preferences

Consumer preferences explain how a consumer would rank a collection of goods (i.e., bundles) or prefer one to another, assuming the goods were available at no cost [13]. Understanding consumer preferences is essential for firms to keep their businesses going. There are different methods to elicit consumer preferences, such as market surveys, auctions, etc. [21]. The main objective of these methods is to identify the most important characteristics of a product to its consumers [88].

Consumers express their preferences through product feedback or purchase decisions. The recent growth of social networks or e-Commerce websites has provided a large volume of feedback data, mostly in two forms of ratings or reviews. Each form conveys different information. Ratings, on one hand, reflect consumers' preferential orders among products. For example, 5-starred products are more preferred to 3-starred ones. Reviews, on the other hand, reveal more details on consumer preferences, beyond what can be expressed by numeric values. For example, consumers would share their experience in using the products, or which features impress them the most, etc. Both rating and review data therefore are useful in learning consumer preferences.

If ratings and reviews capture how consumers enjoy a product, willingness

to pay encompasses their purchase decision. By definition, willingness to pay is the maximum amount that a consumer is willing to give up in exchange for a product. For example, if a consumer u's willingness-to-pay for product A is \$12, it means that u is willing to pay at most \$12 for product A. Any higher price than \$12 would lead to no purchase from u. By $w_{u,A} \geq 0$ we denote u's willingness-to-pay for A. The unit can be any measurement of value, such as dollars. As willingness-to-pay data is often proprietary, in Chapter 5, we discuss how to estimate this quantity from online reviews.

Diversity of Preferences

Consumers have diverse preferences. It is common for consumers to have their own favorite products, as well as unfavorable ones. Knowing the diversity of preferences is important for firms to cater more personalized products or services regarding each individual's tastes. In return, consumers have better experiences in consuming the products, resulting in more value-added benefits for firms such as profit gain, better brand awareness.

Diversity of preferences can be examined from different perspectives. Based on interactions between consumers and products, we propose four different forms of diversity of preferences. Let us consider a preference matrix in which each row corresponds to a consumer, each column corresponds to a product, and each entry contains a real value (e.g., ratings), indicating how much a consumer enjoys a product. The four forms of diversity are combinations of two perspectives, namely dimension of diversity (consumers or products) and level of diversity (individuals or pairs), as illustrated in Figure 1.1.

For the diversity at individual level, the variance of entries within a row (or a column) would signal the diversity of preferences of the corresponding consumer (or product). For instance, the product A in Figure 1.1 receives mixed reviews from consumers. Modeling diversity of individual preferences aims to derive consumer-specific or product-specific models from the preference matrix, such as aspect model [39, 40], matrix factorization [57].

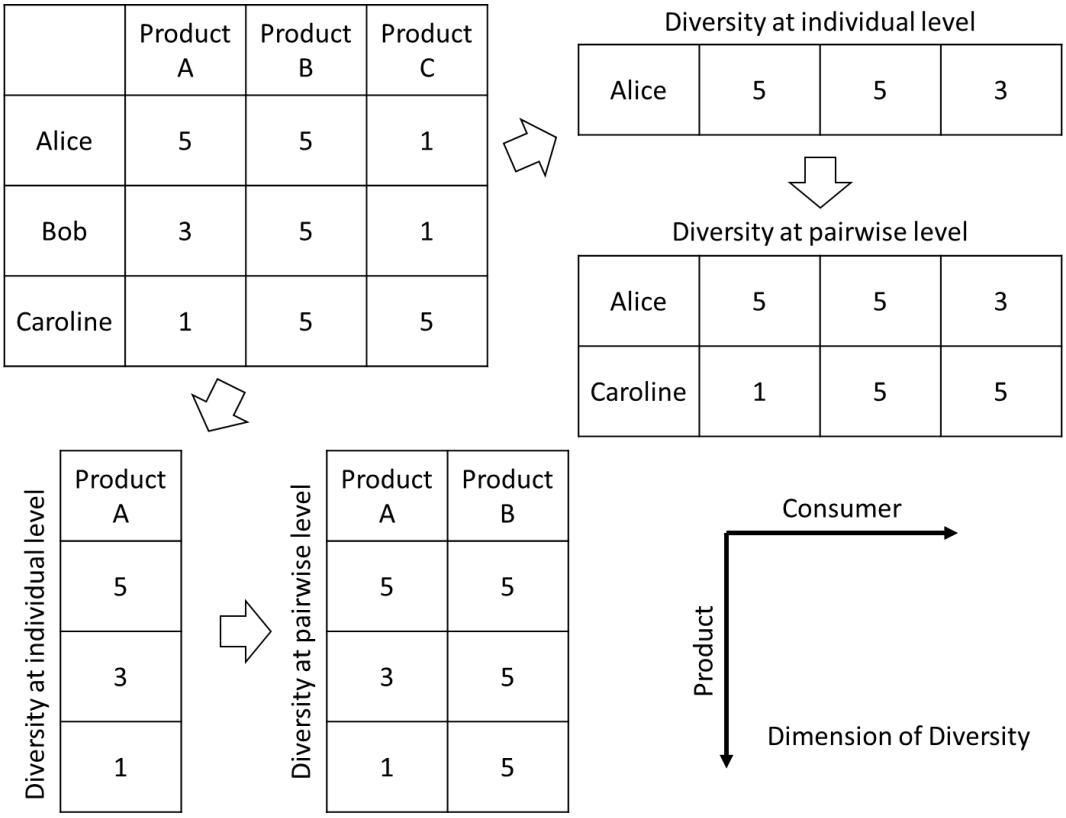


Figure 1.1: Illustration of various forms of diversity in consumer preferences.

For the diversity at pairwise level, the correlation of entries between two rows (or two columns) would indicate the diversity of preferences between the corresponding consumer pair (or product pair). For example, two consumers may share similar ratings on their co-rated products. The degree of correlation would depend on the diversity of each individual's preferences. Existing work in the literature captures the correlation by a single numerical value, which is used to indicate the similarity between objects in recommendation systems, for example, recommending friends, updating news feed [98, 85].

1.1 Research Scope

In this manuscript we study the diversity of preferences between pairs of consumers (or products) in two applications, namely recommendation systems and product bundling. For recommendation systems, we examine the diverse preferences between two consumers on the product basis to contextualize the

Table 1.1: MovieLens users u_{38} and u_{197}

Movie	u_{38}'s rating	u_{197}'s rating	u_{38}'s rating − u_{197}'s rating
Conan the Barbarian	5	1	4
Volcano	5	2	3
First Knight	5	2	3
Scream	5	3	2
G. I. Jane	5	3	2
George of the Jungle	3	1	2
Titanic	5	4	1
Liar Liar	5	4	1
Top Gun	5	4	1
Braveheart	5	5	0
Jurassic Park	5	5	0
Conspiracy Theory	4	4	0
Die Hard (1995)	2	3	-1
Full Metal Jacket	2	3	-1
The Fugitive	3	5	-2
Batman (1989)	1	3	-2
The Godfather	2	5	-3
Die Hard 2 (1990)	1	4	-3
Ben Hur	1	5	-4
The Terminator	1	5	-4

pairwise diversity (Section 1.1.1). For product bundling, we focus on the negative correlation in preferences between two products across consumers to design profitable bundles (Section 1.1.2). Below, we first give an overview of each form of diversity, before motivating specific issues addressed in this dissertation.

1.1.1 Diversity of Preferences Between Consumers Across Various Products

As mentioned above, diversity of individual preferences may lead to diversity of preferences among pairs of consumers across products. To illustrate this point, we use a real-life example from the MovieLens[1] dataset.

Example. In Table 1.1, we show the ratings by two consumers (identified by their anonymized IDs u_{38} and u_{197} respectively) on twenty movies that both of them had rated. The ratings are from 1 (low) to 5 (high). In addition to the

[1]http://grouplens.org/datasets/movielens/

ratings by the consumers on each movie, we indicate their rating differences. The top few movies in the list (e.g., Conan the Barbarian) are movies to which u_{38} assigns high ratings, but u_{197} assigns low ratings, yielding large positive rating differences. The movies in the middle (shaded rows) are those that u_{38} and u_{197} tend to agree on, such as when both like the same movie (e.g., Jurassic Park). The movies at the bottom of the list are movies that u_{38} dislikes, but u_{197} likes (e.g., Ben Hur), yielding large negative rating differences.

Knowing how two consumers are diverse in their tastes can be useful for inferring the preferences of each individual. Indeed, one common approach to capture consumer preferences in the literature is *individual preference* modeling, which is to derive consumer-specific models from their response data. There are several well-known methods, such as aspect models [39, 40], matrix factorization [57], and content-based models [2, 78]. A common challenge of these methods comes from 'cold-start' consumers or products due to their insufficient record of activities (e.g., ratings) for learning accurate models. However, the limited record may already be sufficient to relate their similarity in preferences to another consumer with longer record or more accurate model. This observation underpins the following *shared preference* approaches.

In the *shared preference* modeling approaches, measuring similarities in preferences between two consumers plays the key role. For example, the nearest neighbor-based methods [47, 84, 20] would recommend to a consumer a product which is adopted by most of other consumers sharing similar tastes with him/her. The widely-used similarity measures in literature, such as Cosine Similarity and Pearson correlation coefficient [71], only return a single value, which is implicitly assumed to indicate a certain degree of agreement on *any products* between two consumers. If the value is high, two consumers are considered to have mostly identical preferences, ignoring the fact that they may disagree in other products.

Let us reconsider the example in Table 1.1. Evidently, u_{38} and u_{197} agree

on some movies, and yet disagree on other movies. However, the traditional approach of shared preference is to measure the overall similarity between the two consumers. Let's consider Pearson correlation and Cosine similarity as the similarity measure in this example. These similarity measures are defined later in Section 4.1. Pearson correlation returns a similarity score of -0.25 on the scale of $[-1, 1]$, indicating slight differences. Under Cosine similarity, the score is 0.81 on the scale of $[0, 1]$, indicating very high similarity. Hence, these two different similarity measures yield very different conclusions. This drives home the point that a single value cannot capture the varying preferences of consumers. Therefore, agreement on preference should be seen in the context of individual products.

Problem. Given a consumer set, a product set, and some product ratings by consumers, we seek a model to capture the agreement between a pair of consumers on a specific product. One key observation is that the observed rating values provide signals of the agreement or disagreement. As suggested by Table 1.1, the two consumers are likely to disagree when their co-ratings differ. Modeling the agreement in preferences gives rise to two sub-problems. The first is how to express the agreement and disagreement in a more principled fashion. Rather than having yet another real-valued similarity, we propose to adopt a probabilistic modeling, expressing the probability that two consumers agree on a specific product. The second is that not all rating differences are observed, arising directly from not having observed all possible ratings. Therefore, there is also a need to predict the "unseen" rating differences.

Scope. As one of the most prominent applications of modeling consumer preferences, we study diversity of preferences among consumers in the context of recommendation systems. Recommendation systems are systems that match recommendations in a certain context (e.g., friends in social networks, products in online retailers, etc.) with consumer preferences. In recent years, recommendation systems have been prevalent in online platforms such as so-

cial networks or e-retailer websites for their value-added benefits. According to Xavier Amatriain, the ex-Research/Engineering director at Netflix, "On Netflix, 2/3 of the movies watched are recommended. Recommendations on Google News generate 38% more clickthrough. 35% sales on Amazon come from recommendations."[2] Improving the quality of recommendations therefore has drawn a lot of attention from both industry and academia.

While our work is related to recommender systems, our focus is on modeling agreement in preferences, and not on rating prediction. We only assume the availability of rating data in our work. Moreover, we also assume that ratings are truthful and reflective of consumer preferences (and not artefacts of dishonesty or fraud [30]), which we believe is true for most consumers.

1.1.2 Diversity of Preferences Between Products Across Various Consumers

Similar to consumer pairs, two products can be valued similarly or differently from a consumer. A portion of population may like or dislike both products. The others may have mixed reviews. Since this study involves consumer's purchase decisions, we consider willingness-to-pay instead of ratings as a proxy to consumer preferences. Let us consider a synthetic example as follows.

Example. Assume a firm wants to sell its two flagship products A and B to a market of three consumers u_1, u_2 and u_3. The variable cost for each product is \$1. Also assume that each consumer's willingness-to-pay on every product is given as described in Table 1.2.

	willingness-to-pay	
Consumer	$w_{u,A}$	$w_{u,B}$
u_1	\$11.00	\$4.00
u_2	\$8.00	\$3.00
u_3	\$5.00	\$11.00

Table 1.2: Example on diversity in consumer valuations between two products

[2] http://www.slideshare.net/xamat/recommender-systems-machine-learning-summer-school-2014-cmu

In Table 1.2, $w_{u,A}$ and $w_{u,B}$ represent willingness-to-pay of a consumer u on A and B respectively. Among three consumers, u_1 and u_2 both value A higher than B. On the contrary, u_3 has higher valuation on B than A. Hence, u_3 is said to have negative correlation in preferences with u_1 and u_2.

Scope. We study the notion of diversity in preferences among products in the context of product bundling. *Product bundling*, or selling two or more items for one price, is a pervasive marketing strategy across many industries. Comcast and many other cable television companies worldwide sell subscriptions not only for individual channels, but also for bundles of channels. Telecommunication providers, such as AT&T, frequently offer a bundle of services, including cable, telephone, and Internet subscriptions. Travel packages commonly bundle airfare, hotel stay, and attractions. Restaurants often offer fixed price menus.

Product bundling brings benefits to both sellers and customers. It takes advantages of variance in customer willingness-to-pay, i.e., different consumers are willing to pay different products at various prices, to attract them to purchase more products. Hence, it also increases sales revenue for firms. In the example in Table 1.2, by selling two items A and B separately, the firm would "miss" both consumers u_1 and u_3 as their valuations on two products are negatively correlated. Indeed, when $p_A = \$5$, all three consumers would purchase A, leading to a profit of $3 \times (\$5 - \$1) = \$12$. Similarly, we obtain \$14 and \$11 in profit by setting the price of p_A at \$8 and \$12 respectively. Hence, the optimal price for p_A is \$8 which yields the maximum profit. Following the same process for B we can arrive at $p_B = \$11$. u_1 cannot afford B, as well as u_3 cannot afford A.

As shown in Table 1.3, this negative correlation is subsumed when A and B are sold as a bundle, which attracts more purchases and leads to higher profit. When we set the price of the bundle $p_{AB} = \$16$, both u_1 and u_3 can afford the bundle, leading to a higher profit of $2 \times (\$15 - \$2) = \$26.00$ than selling A

Consumer	willingness-to-pay			Components		Bundle
	$w_{u,A}$	$w_{u,B}$	$w_{u,AB}$	$p_A = \$8.00$	$p_B = \$11.00$	$p_{AB} = \$15$
u_1	\$11.00	\$4.00	\$15.00	✓		✓
u_2	\$8.00	\$2.00	\$10.00	✓		
u_3	\$5.00	\$11.00	\$16.00		✓	✓
Profit				*\$24.00*		*\$26.00*

Table 1.3: Positive Example of Exploiting Diversity in Consumer Preferences

and B separately.

Problem. As it is impractical to offer all possible bundles to market, it is important for firms to determine from consumer demands which products to be sold as bundles, and at what price to maximize their business targets. Depending on business situations, firms may seek different configurations of bundles. When firms want to offer all its products to the market, to avoid competition among bundles, a desired solution is a partition of a given product set. In scenarios where firms are only interested in the few most profitable bundles, top-K bundles are a suitable solution. Note that top-K bundles do not necessarily form a partition. We call this problem in general the *bundle set design problem*. The problem poses computational challenges as the number of possible bundles in the search space would grow exponentially relative to N. Moreover, to obtain the desired bundle set, we further need to consider various combinations of these bundles, increasing the complexity of the problem.

1.2 Contributions and Organization

After an overview on consumer preferences in Chapter 2, we organize the remaining chapters into two parts in regard to each diversity perspective. Below, we list our contributions in the same order as their corresponding chapters.

Part I. Diversity of preferences between consumers across products

1. A predictive model for rating differences between two consumers

Previous latent-factor based models focus on predicting ratings. Although we can use estimations from these models to compute rating differences, they

may not capture diversity across rating differences as it is not their objective. In Chapter 3, we propose *Differential Probabilistic Matrix Factorization* or *DPMF*, a latent-factor based model, to capture "seen" and predict "unseen" rating differences between two consumers on any products.

2. A contextual similarity measure

The most common existing similarity measures are not informative enough to differentiate similarity between a pair of consumers on a product-by-product basis. To fit in the gap, we propose modeling product-specific context in estimating the agreement between two consumers. To realize this modeling, we develop a probabilistic generative model in Chapter 4, called *Contextual Agreement Model* or *CAM*, based on Gaussian mixtures. We enforce a monotonicity property that results in a specific parameter constraint, and describe how to learn the constrained parameters with Expectation Maximization.

Part II. Diversity of preferences between products across consumers

1. A framework to extract willingness-to-pay from online reviews

Willingness-to-pay data is often proprietary. Traditional willingness-to-pay estimation methods in the literature are unscalable due to logistical costs, e.g., the cost to conduct survey, or to set up the laboratory experiments [21]. In Chapter 5, we propose a framework to automatically extract these hidden values from online reviews for its public availability at large volume. The framework consists of four components. The first component identifies product features mentioned in a given review. The second component detects how consumer values each mentioned features. The third component employs an existing method, so called Conjoint Analysis, to elicit consumer's utility from their valuations on features. The fourth component transforms the elicited utility into willingness-to-pay.

2. Efficient algorithms for finding profit-maximizing bundling configurations

In Chapter 6, we formulate the *bundling configuration* problem, in which we

seek a partition of a given set of products into bundles, such that the total profit of selling each bundle is maximized. We prove the problem is NP-Hard by a reduction from 3-uniform hypergraph matching. To solve the problem, we propose two efficient heuristic algorithms to find profit-maximizing bundling configurations. Both algorithms follow the bottom-up style in hierarchical clustering. At every step, the first algorithm find the most profitable bundling configuration by pairing existing components, which can be single products or bundles. The second algorithm, on the contrary, only forms the most profitable bundle in each step. Despite of being heuristic, we also show that both algorithms provide comparable results compared to the optimal solution.

3. Efficient algorithms for finding the profit gain maximizing top-K diverse bundles

In Chapter 7, we propose a novel top-K problem based on the concept of bundling. In particular, we seek a diverse bundle set of K bundles with maximum profit gain. The problem is proved to be NP-Hard. We propose a two-phase greedy algorithm to find the profit gain maximizing top-K diverse bundles. In the first phase, bundles with positive profit gain are generated. In the second phase, a set of K diverse bundles with maximum total profit gain is selected. The algorithm returns more profit gain bundle sets than the ones from Chapter 6.

Chapter 2

Overview of Consumer Preferences

In this chapter, we give an overview of prior work related to consumer preferences. After introducing relevant fundamental concepts, we first summarize a set of approaches to model consumer preferences, before listing out a couple of applications which are related to our study.

The following definitions are used throughout the dissertation.

Consumer feedback

Consumer interactions on products in online settings. There are two types of feedback, namely direct preference-stated feedback (e.g., ratings, reviews, etc.) and indirect preference-stated feedback (e.g., clicks, number of view, etc.). As mentioned in Chapter 1, our study focuses on online ratings and reviews.

Cold-start consumers

Consumers with little to no feedback. In the online systems, these consumers are infrequent or new visitors to the systems. Modeling their preferences accurately is challenging.

Willingness-to-pay

The maximum amount of money a consumer is willing to pay for one unit of the product.

Cost

In Economics, a product is associated with two main types of costs, namely fixed cost and variable cost. Fixed cost indicates expenses that are independent of the production volume. Meanwhile, variable cost is the product cost to produce an additional unit of the product. Only variable cost is affected by the quantity sold (which in turn is affected by pricing).

Profit

The profit of selling a product at price p is computed as follows

$$profit \propto quantity\ sold \times (p - variable\ cost)$$

If each consumer only wants a unit of the product, the quantity sold is the number of consumers purchasing the item. When the variable cost is zero or very small, such as in certain markets like digital goods (e.g., cable TV, video on demand) [10], profit maximization is equivalent to revenue maximization. If "profit" is mentioned without a specific price, we mention the optimal profit firm can earn from the product.

Surplus and deadweight loss

If a consumer purchases a product, surplus is defined to be the difference between his or her willingness-to-pay and the product's price. If the consumer cannot afford the product, the surplus is zero. His or her willingness-to-pay in that case is called deadweight loss.

2.1 Modeling Consumer Preferences

It has been well-established in the Economics literature that consumer preferences on a product/service can be measured by a value called utility [13]. The utility reflects the degree of "satisfaction" or "pleasure" for consuming the product. In practice, the quantity is conveyed in many forms of product feedback. An example would be product ratings in which high ratings imply

high utility and vice versa. In the case of relative choice, products with higher utility are more preferred to ones with lower utility. A consumer is also willing to pay more for products giving her higher utilities.

Conceptually, modeling consumer preferences aims to *explain or estimate how much a consumer enjoys a product*, or the utility obtained by him or her from consuming the product. This utility is often assumed to be a summation of the utilities obtained from each product feature or characteristic [59]. Let us consider a consumer u and a product i of K features. Each k-th feature is associated with two values, namely the *feature coefficient* q_{ik} and *consumer valuation* s_{uk}. The overall utility $U(u, i)$ is hence defined as follows [63]

$$U(u,i) \propto \sum_k s_{uk} \times q_{ik} = \boldsymbol{S}_u^T \boldsymbol{Q}_i \tag{2.1}$$

In Equation 2.1, the feature coefficient vector $\boldsymbol{Q}_i$ represents the intrinsic features of the product itself, and $\boldsymbol{S}_u$, the vector of feature valuations, reflects how the consumer in consideration values the features. For each feature k, the product between s_{uk} and q_{ik} is often known as **part-worth utility**. High part-worth utilities often come from the consumer's favorite features. For example, a consumer who prefers mobile phones with large screens would have high part-worth utility on the "screen size" feature on phones with large screens.

Given both $\boldsymbol{S}_u$ and $\boldsymbol{Q}_i$, we can explain or estimate how much a consumer would prefer a product using Equation 2.1. However, in practice, either or both of the vectors are often missing. We only observe consumer feedback on products. Hence, the crux of modeling consumer preferences is to recover the missing parameters from the observed feedback. Since $\boldsymbol{Q}_i$ can be sometimes extracted from product metadata, two common contexts in the literature are 1) $\boldsymbol{S}_u$ is missing and 2) both vectors are missing. Below, we survey related work in these contexts, which is relevant to our study in the subsequent chapters.

2.1.1 Modeling latent $\boldsymbol{S}_u$

Modeling consumer preferences when $\boldsymbol{S}_u$ is missing is often found in Conjoint Analysis, a widely-used method to elicit consumer utility or willingness-to-pay using survey data [46, 22, 29]. Basically, the participants are asked to rank several product profiles with variant features, for example cameras with different colors or battery life. The modeling objective is to align the utility in Equation 2.1 for every pair of consumer and product with the observed relative preferences. Specifically, since $\boldsymbol{Q}_i$ is determined through i's profile, $\boldsymbol{S}_u$ is estimated from her response data using regression analysis [46] or optimization algorithm [29]. More details can be found in Chapter 5 where we use the similar approach to elicit consumer willingness-to-pay from online reviews.

2.1.2 Modeling latent $\boldsymbol{S}_u$ and $\boldsymbol{Q}_i$

Modeling consumer preferences when both $\boldsymbol{Q}_i$ and $\boldsymbol{S}_u$ are missing draws more attention from researchers. The main reason is that it is not always plausible to derive $\boldsymbol{Q}_i$. First, survey methods are often costly to carry out, hence, not applicable to large scale. Second, a fixed $\boldsymbol{Q}_i$ extracted from product descriptions may not cover all aspects of consumers' interests such as ease of use, consumer services, etc. Third, it is not straightforward to extract $\boldsymbol{Q}_i$ for some product categories like food, books, and so on. Therefore, a common remedy in the literature is to treat $\boldsymbol{Q}_i$ as hidden parameters, and optimize $\boldsymbol{S}_u$ and $\boldsymbol{Q}_i$ to fit observed data.

Depending on the type of observed data, the modeling objective may vary. One objective could be to preserve the ranking among the products [46, 29]. Another objective is to fit the observed ratings [76, 87, 57]. Below, we survey related work on modeling preferences, first focusing on individual consumers, and then in the role of context.

Individual preference. The main step in modeling individual preference is to construct a preference model for each consumer, which is then used to de-

rive predictions. We focus on two popular modeling directions, namely content-based and collaborative-based. Each has different data focus. To be specific, content-based directions look at backgrounds of consumers or products to derive estimations. Meanwhile, collaborative-based approaches concentrate on consumer feedback.

The underlying assumption of the content-based individual preference modeling [2, 78, 68] is that consumer preferences can be determined based on content (i.e., meta-data, text, etc.) Specifically, $\boldsymbol{S}_u$ is modeled as a content vector whose dimensionality is the vocabulary size (e.g., $tf \cdot idf$ vector), derived from the content of products that u likes. Each word in the vocabulary represents a product's feature. Similarly, $\boldsymbol{Q}_i$ is the $tf \cdot idf$ vector derived from product i's content. The utility $U(u, i)$ is high if the product i has similar description with the products liked by u.

One drawback of this approach is that it does not really take into account diversity of consumer preferences since disliked products are often ignored. Besides, the underlying assumption suggests a strong connection between content and consumer preferences, which may not always be the case. Plus, shortage in profile information also affects the quality of representation vectors [77]. These limitations motivate utilizing feedback data to make recommendations.

The most popular collaborative-based approach to model individual preference is probably the *latent factor-based models*, which assume both feature coefficient and feature valuation vectors (Equation 2.1) are not observed, and can be derived from feedback data. There are two seminal models in this direction, namely *aspect model* [39, 40] and matrix factorization [57]. In the *aspect model*, $\boldsymbol{S}_u$ is modeled as a probability distribution $\{\mathrm{P}(z_k|u)\}_{k=1}^{K}$ over K latent aspects. Each aspect z_k, or product feature k, has a distribution over products i to be adopted, i.e., $\mathrm{P}(i|z_k)$, or ratings r, i.e., $\mathrm{P}(r|z_k, i)$. Hence, each coefficient vector $\boldsymbol{Q}_i$ also has K dimensions, containing the adoption probabilities $\mathrm{P}(i|z_k)$ for all aspects.

For the *matrix factorization*, u's preference is modeled as a column vector $\boldsymbol{S}_u$ in a K-dimensional latent space. Similarly, i is associated with a rank-K column vector $\boldsymbol{Q}_i$. The rating prediction $\hat{r}_{ui}$ is given by ${\boldsymbol{S}_u}^T\boldsymbol{Q}_i$. There are different methods [61, 96, 60, 72], which vary in their objective functions, including probabilistic variants [76, 87].

Since these models considered both liked and disliked products, they also capture diversity in preferences on different consumers across products. These models are also advantageous to incorporate additional domain knowledge into the learning phase to improve estimation on "cold-start" cases [71, 70, 91, 6].

Contextualized preference. Most works described above base their approaches on the dyad of user-item pair. In some cases, additional information or "context" may be available. Rather than pairs $\langle u, i\rangle$, we observe triplets $\langle u, i, c\rangle$ where c refers to some context such as time [112, 56], location [62], tags [83], etc. Here, we briefly describe two common approaches to dealing with triplets, i.e., incorporating contextual information into a preference model. A broader overview can be found in [3].

The first direction is to separate the ratings of different contexts, and model each context independently. For example, suppose each day of the week is a separate context, we could build seven matrix factorization models corresponding to every day of the week [56]. [62] considers a similar strategy but uses locations as contexts. Let us denote context as c, Equation 2.1 can be slightly modified to incorporate c as follows

$$U(u, i, c) \propto \boldsymbol{S}_{u,c}^T\boldsymbol{Q}_{i,c}, \tag{2.2}$$

where $\boldsymbol{S}_{u,c}$ and $\boldsymbol{Q}_{i,c}$ are context-specific model parameters. While being applicable to those cases, this approach is not suitable for our problem, because we are interested in individual items as contexts.

The second direction is to model triadic relationships $\langle u, i, c\rangle$ directly via another form of factorization called tensor factorization. The basic form of

tensor factorization is *Tucker Decomposition* [102]. To improve efficiency, several works propose special forms of tensor decomposition, such as *Canonical Decomposition* (CDTF) [112] and *Pairwise Interaction Tensor Factorization* (PITF) [83]. The main difference between this direction and the above is that each context c is also associated with a vector $\boldsymbol{T}_c$ of K-dimensions as shown below

$$U_{\text{CDTF}}(u, i, c) \propto \sum_k s_{uk} q_{ik} t_{ck} \tag{2.3}$$

$$U_{\text{PITF}}(u, i, c) \propto \boldsymbol{S}_u^T \boldsymbol{Q}_i + \boldsymbol{S}_u^T \boldsymbol{T}_c + \boldsymbol{Q}_i^T \boldsymbol{T}_c \tag{2.4}$$

In Chapters 3 and 4, we study contextualized preference on the product basis. While tensor factorization cannot solve our problem directly, in Section 3.2.1, we show how it may be adapted into a sub-component of our model, with a specific modification to fit our scenario of modeling rating differences.

2.2 Applications of Consumer Preferences

The concept of consumer preferences finds itself in many applications across various domains and industry. Below, we review several essential applications of consumer preferences for businesses. Our review is not intended to be exhaustive; we include only what we believe to be some of the chief development in the area and most importantly, related to our study.

Knowing consumer preferences is necessary for firms since it helps to explain consumer behaviors, such as how they make decisions among alternatives (e.g., brands, retailers), which factors are the most important for them, etc. Based on that, firms can make important decisions to their businesses such as product design, pricing, marketing strategies. Other than traditional settings, consumer preferences is also used extensively in online platforms such as social media platforms or e-Commerce websites.

Understanding Consumer Behavior

Consumer behavior is "the study of the processes involved when individuals or groups select, purchase, use, or dispose of products, services, ideas, or experiences to satisfy needs and desires"[95]. The study has covered a lot of grounds such as environment[109], public transportation systems[32], social science[86], etc. For example, in [109], the authors study the role of various factors, e.g., price, species, certifying agency, knowledge and perceptions of the status of fish stocks, in consumer choices between eco-labeled seafood and unlabeled one. In the context of consumer choices on alternative products, consumer behavior is often explained by consumer utility. Basically, consumers are often assumed to choose products maximizing their utility within their budget constraints [13].

Deriving Product Price

Based on market demands, represented by consumers' willingness-to-pay, firms can determine the profit maximizing price [73]. A straightforward pricing strategy is an exhaustive search on all the willingness-to-pay values to find the optimal price, which is illustrated in Section 1.1.2. Besides maximizing the profits, there are other pricing objectives [82]. We discuss more details about different pricing objectives in Chapter 5.

Enhancing Product Design

An application of part-worth utilities in product design is to select features for a set of products [55]. Given a set of feature valuation vectors $\boldsymbol{S}_1$, $\boldsymbol{S}_2$, ..., the application is to determine $\boldsymbol{Q}_i$ for each product so as to maximize the total utility obtained by consumers. For example, [69] studies how consumer values food labeling. [31] focuses on consumer preferences for wine attributes.

Another application of consumer preferences in product design is bundle design as mentioned in Section 1.1.2. The subtle difference between bundle design and product line design is that the former application also includes a pricing problem, meanwhile the latter does not. However, both applications are shown to be NP-Hard problems, hence requiring some efficient heuristics.

As our focus is on the bundle design, we provide more comprehensive survey on product bundling in general and bundle design in particular in Chapter 6.

Devising Marketing Strategies

Another application of knowing $\boldsymbol{S}_u$ is to help marketers to adjust their marketing strategies. For example, they can emphasize the features valued the most by the consumers, i.e., high $\boldsymbol{S}_u$'s entries, in their advertising campaigns. Or they can cater the campaigns to specific consumer groups. Moreover, firms can leverage consumer preferences to position their products [52], to measure the impact of loyalty programs on consumer purchase behavior [67].

Recommending on Online Platforms

Utility estimation finds itself in many applications of recommendation in online platforms such as social media and e-Commerce. Social media platforms are websites allow users to create, share and exchange content with three main purposes: 1) networking (e.g., Facebook) 2) promoting user content (e.g., Google) and 3) sharing (e.g., Delicious). One of the common challenges of these platforms is to maintain visitors' retention. To do so, identifying the users' preferences and suggesting items which draws their attentions become essential tasks. Some common recommendation tasks are video recommendation [113], friend recommendations [15], social feeds recommendation [94], tags recommendations [92], to name but a few.

For e-Commerce websites, the most important goal is to retain and increase the number of purchases in their platforms. Besides appealing promotions, good customer services, product recommendation also helps in increasing more purchases. There is a plethora of work in the literature about improving the product recommendation services in many aspects, mostly focusing on the quality of recommendations. Our study in Part I of this manuscript also contributes to this research direction.

Part I

Diversity of Preferences between Consumers across Various Products

Chapter 3

Modeling Differences in Responses

3.1 Overview

As discussed in the previous chapters, knowing individual preferences is important as it tells us in what aspects an individual enjoys a product. We posit that understanding differences in preferences among individuals is also equally essential. First and foremost, it helps to provide better recommendations as two consumers may not always agree on the same products. Predicting products which they would have opposite views can preclude unwanted recommendations. In addition, it can be useful in social networks to provide better friend recommendations [98]. Third, it helps to characterize preferences of various consumer groups based on demographics or their background.

In this chapter we consider ratings as a form of utility truly representing consumer preferences. Intuitively, difference between two rating values signals the agreement or disagreement. For example, large difference in ratings would likely imply the two raters disagree with each other. Since rating differences are not fully observed, the main focus of this chapter is model and estimate rating differences between two consumers.

3.1.1 Problem Statement

Notations. The universal set of consumers is denoted as $\mathcal{U}$, and we use u or v to refer to a consumer in $\mathcal{U}$. In turn, we use i or j to refer to an item in the universal set of products $\mathcal{I}$. The rating by u on i is denoted as r_{ui}. The set of all ratings observed in the data is denoted R. We seek to model consumer-consumer-item triplets $\langle u, v, i \rangle$. The universal set of triplets comprises $\mathcal{U} \times \mathcal{U} \times \mathcal{I}$, excluding triplets involving the same consumers, e.g., $\langle u, u, i \rangle$. Each triplet $\langle u, v, i \rangle$ is associated with a random variable x_{uvi}, which are essential to our probabilistic modeling.

The variable $x_{uvi} \in \mathbb{R}$ is real-valued. It represents the indicator of agreement between u and v on i, some of which are observed in the data. x_{uvi} can be expressed as a function of ratings, i.e., $x_{uvi} = \mathcal{F}(r_{ui}, r_{vi})$. While there are many possible definitions of $\mathcal{F}$, in this paper, we simply use the *rating difference* between two consumers on the same item, as shown in Equation 3.1.

$$x_{uvi} = r_{ui} - r_{vi} \tag{3.1}$$

This choice of function also implies the relationship $x_{uvi} = -x_{vui}$.

Problem Formulation. Given rating data R, and the above x_{uvi} definition, x_{uvi} is not observed if either $r_{ui} \notin R$ or $r_{vi} \notin R$. Let's denote $\hat{x}_{uvi}$ as unobserved *rating difference*. We seek to estimate all $\hat{x}_{uvi}$'s from R.

One insight is that the $\hat{x}_{uvi}$'s are not independent from one another. All triplets involving the same product i or the same consumer pair (u, v) will share some dependency. Furthermore, the triplet also captures the interaction between consumers and products. Our approach is to model the generation of x_{uvi} based on consumer- or product-specific parameters to generate/predict unseen $\hat{x}_{uvi}$ through matrix or tensor factorization in Section 3.2.

3.2 Methodology

In this section we show how to estimate the unseen triplets $\hat{x}_{uvi}$ from rating data. Specifically, we first introduce methods directly estimating $\hat{x}_{uvi}$ from observed x_{uvi} in Section 3.2.1, namely *Pairwise Probabilistic Matrix Factorization (PPMF)*, *Canonical Decomposition Tensor Factorization (CDTF)*, and *Pairwise Interaction Tensor Factorization (PITF)*. These rating difference-based methods do not explicitly capture the formulation of rating difference, hence may not well predict $\hat{x}_{uvi}$. Since rating differences are constructed from ratings, we then introduce methods to estimate unseen ratings from the given rating data in Section 3.2.2, *Probabilistic Matrix Factorization (PMF)* and *Differential Probabilistic Matrix Factorization (DPMF)*. The estimated ratings are later used with Equation 3.1 to predict the unseen rating differences. The distinctions among methods are depicted in Figure 3.1. Note that in these methods, we do not incorporate bias terms [57] (an orthogonal issue), in order to isolate the effects of the structure of the matrix and tensor factorizations.

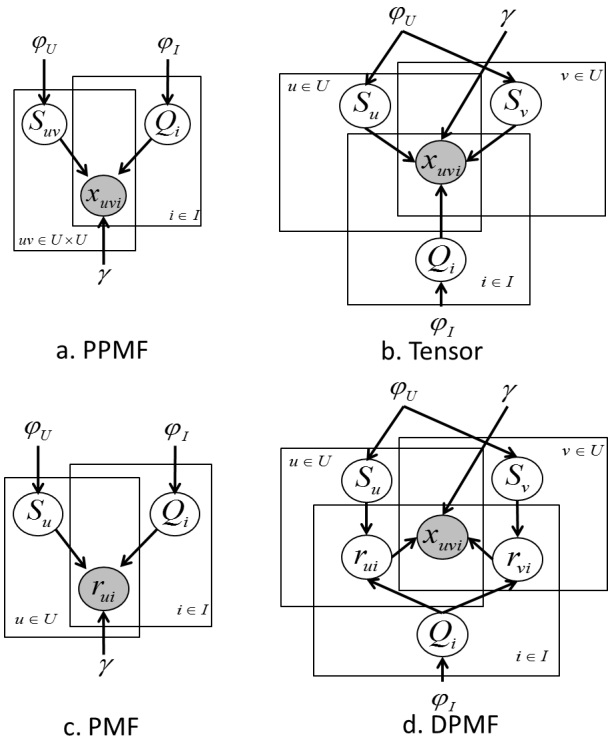


Figure 3.1: Plate Diagrams: Factorization Models for Rating Difference Prediction

3.2.1 Factorizing Rating Differences

A straightforward approach to estimate unseen triplets $\hat{x}_{uvi}$ is to learn a model capturing the observed rating differences. Here we investigate two different views in this direction. The first view represents the triplets as a two-dimensional matrix, with consumer pairs on one dimension and items on the other dimension. Filling the missing entries of the matrix can be done by employing matrix completion techniques such as matrix factorization [76]. By assigning each consumer-pair a set of parameters, this approach would demand a significant number of parameters when the number of consumers is large, posing challenges in learning the model effectively and efficiently.

The second view represents the triplets as a three-dimensional $|\mathcal{U}| \times |\mathcal{U}| \times |\mathcal{I}|$ tensor, and employs tensor factorization to predict the missing entries. This approach does not assign parameters to every consumer pair, potentially reducing the number of parameters in comparison to the matrix factorization approach.

Matrix Factorization-based approach. In the first view, the main objective is to fit a matrix $\mathbb{X}$, of size $|\mathcal{U}|^2 \times |\mathcal{I}|$. Each row corresponds to a pair of consumers uv. Each column relates to an item i. Each element x_{uvi} is the rating difference $r_{ui} - r_{vi}$. To approximate $\mathbb{X}$, we associate each consumer pair with a rank-K vector S_{uv}, and each item with Q_i. To generate $\hat{x}_{uvi}$, we draw it from a Normal distribution, as in Equation 3.2.

$$\hat{x}_{uvi} \sim \mathcal{N}({S_{uv}}^T Q_i, \gamma^2) \tag{3.2}$$

We call this approach *Pairwise PMF* or *PPMF*. The plate diagram is shown

in Figure 3.1(a). The objective function of *PPMF* is specified in Equation 3.3.

$$E = \frac{1}{2} \sum_{uv \in \mathcal{U} \times \mathcal{U}, u \neq v} \sum_{i \in \mathcal{I}} \mathbb{I}_R(u, v, i)(x_{uvi} - {S_{uv}}^T Q_i)^2 + \frac{\lambda_U}{2} \sum_{uv \in \mathcal{U} \times \mathcal{U}, u \neq v} ||S_{uv}||^2 + \frac{\lambda_I}{2} \sum_{i \in \mathcal{I}} ||Q_i||^2 \tag{3.3}$$

The estimation is done using gradient descent, with the following gradients. Once the parameters are learned, we then predict each $\hat{x}_{uvi}$ as ${S_{uv}}^T Q_i$.

$$\frac{\partial E}{\partial S_{uv}} = -(x_{uvi} - {S_{uv}}^T Q_i)Q_i + \lambda_U S_{uv} \tag{3.4}$$

$$\frac{\partial E}{\partial Q_i} = -(x_{uvi} - {S_{uv}}^T Q_i)S_{uv} + \lambda_I Q_i \tag{3.5}$$

While *PPMF* estimates $\hat{x}_{uvi}$ directly, it suffers from two design issues. First, it blows up the number of parameters, as we now have to learn the S_{uv} for every pair, instead of every consumer. Second, it assumes that the vectors S_{uv} and $S_{uv'}$ are independent, even as they share the same consumer u. These are somewhat rectified by the tensor factorization approach below.

Tensor Factorization-based approaches. Rating differences $\hat{x}_{uvi}$ can be represented as triadic interactions between two consumers and one item. These triadic interactions can be encapsulated by a three-dimensional tensor. Let $\mathcal{X}$ be the 3-dimensional $|\mathcal{U}| \times |\mathcal{U}| \times |\mathcal{I}|$ tensor. Each element $x_{uvi} \in \mathcal{X}$ is an instance of rating difference as defined in Equation 3.1. We still associate each consumer u with a latent vector $S_u \in \mathbb{R}^K$, and each item i with $Q_i \in \mathbb{R}^K$. Applying the basic *Tucker Decomposition* [102] would require the following factorization, where $C \in \mathbb{R}^{K \times K \times K}$ is the core tensor that reflects the interaction among different components.

$$\hat{x}_{uvi} = \sum_{x=1}^{K} \sum_{y=1}^{K} \sum_{z=1}^{K} C_{xyz} S_{ux} Q_{iy} S_{vz}, \tag{3.6}$$

There are two issues with this factorization. The first issue arises from the

requirement in our scenario that $\hat{x}_{uvi} = -\hat{x}_{vui}$. This conflicts with the commutativity of Equation 3.6, because both S_u and S_v appear in the factorization of both $\hat{x}_{uvi}$ and $\hat{x}_{vui}$. This issue did not arise in the conventional application of Tucker Decomposition [51], because there the three dimensions are separate, whereas two of our three tensor dimensions represent consumers. To resolve this, we would seek to factorize only the magnitudes, which in this case will be the same, i.e., $|\hat{x}_{uvi}| = |\hat{x}_{vui}|$.

The second issue is that of computational efficiency due to the nested sum of degree three in Equation 3.6. This is a known issue in tensor factorization [83]. We resolve this by adapting two existing simplifications of Tucker Decomposition, namely Canonical Decomposition Tensor Factorization (CDTF) [112] and Pairwise Interaction Tensor Factorization (PITF) [83]. In the following, we review these works, and identify the required modification due to the first issue mentioned above.

Canonical Decomposition Tensor Factorization (CDTF). CDTF is a special case of the Tucker Decomposition when the core tensor C is diagonal. This simplification allows us to collapse the nested sum from degree three to degree one, which improves the computational complexity from $\mathcal{O}(K^3)$ to $\mathcal{O}(K)$. $\hat{x}_{uvi}$ is thus modeled as an inner-product of S_u, S_v and Q_i as show in in Equation 3.7.

$$\hat{x}_{uvi} \approx \langle S_u, S_v, Q_i \rangle = \sum_{k=1}^{K} S_{uk} S_{vk} Q_{ik} \tag{3.7}$$

We account for noise in x_{uvi} by introducing a Gaussian prior.

$$\hat{x}_{uvi} \sim \mathcal{N}(\langle S_u, S_v, Q_i \rangle, \gamma^2) \tag{3.8}$$

To learn the parameters, we formulate the following objective function. To deal with the commutativity issue above, we model the absolute value $|x_{uvi}|$.

$$E = \frac{1}{2}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U},v\neq u}\sum_{i\in\mathcal{I}}\mathbb{I}_R(u,v,i)(|x_{uvi}|-\langle S_u,S_v,Q_i\rangle)^2+\frac{\lambda_U}{2}\sum_{u\in\mathcal{U}}||S_u||^2+\frac{\lambda_I}{2}\sum_{i\in\mathcal{I}}||Q_i||^2 \tag{3.9}$$

Due to the symmetric property, each pair of consumers u and v is considered only once for each item i in Equation 3.9. Estimation by gradient descent uses the gradients below, where $\odot$ is the element-wise product between two vectors.

$$\frac{\partial E}{\partial S_u} = -(|x_{uvi}| - \langle S_u, S_v, Q_i\rangle)(Q_i \odot S_v) + \lambda_U S_u \tag{3.10}$$

$$\frac{\partial E}{\partial S_v} = -(|x_{uvi}| - \langle S_u, S_v, Q_i\rangle)(Q_i \odot S_u) + \lambda_U S_v \tag{3.11}$$

$$\frac{\partial E}{\partial Q_i} = -(|x_{uvi}| - \langle S_u, S_v, Q_i\rangle)(S_u \odot S_v) + \lambda_I Q_i \tag{3.12}$$

Pairwise Interaction Tensor Factorization (PITF). Another simplification of Tucker Decomposition is *PITF*, which assumes that $\hat{x}_{uvi}$ is the sum of three pairwise products as follows, which still achieves a computational complexity of $\mathcal{O}(K)$ because it involves three summations of degree one.

$$\hat{x}_{uvi} \approx S_u\odot Q_i+S_v\odot Q_i+S_u\odot S_v = \sum_{k=1}^{K} S_{uk}Q_{ik}+\sum_{k=1}^{K} S_{vk}Q_{ik}+\sum_{k=1}^{K} S_{uk}S_{vk} \tag{3.13}$$

Similarly to the *CDTF*, we also introduce a probabilistic version by modelling Gaussian prior as follows.

$$\hat{x}_{uvi} \sim \mathcal{N}(S_u \odot Q_i + S_v \odot Q_i + S_u \odot S_v, \gamma^2), \tag{3.14}$$

This results in the following objective function. Note that *PITF* also re-

quires the use of absolute value $|x_{uvi}|$ to deal with the commutativity issue.

$$E = \frac{1}{2}\sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U},v\neq u}\sum_{i\in\mathcal{I}}\mathbb{I}_R(u,v,i)(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))^2 + \frac{\lambda_U}{2}\sum_{u\in\mathcal{U}}||S_u||^2 + \frac{\lambda_I}{2}\sum_{i\in\mathcal{I}}||Q_i||^2 \quad (3.15)$$

Estimation by gradient descent uses the gradients below.

$$\frac{\partial E}{\partial S_u} = -(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))(Q_i + S_v) + \lambda_U S_u \quad (3.16)$$

$$\frac{\partial E}{\partial S_v} = -(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))(Q_i + S_u) + \lambda_U S_v \quad (3.17)$$

$$\frac{\partial E}{\partial Q_i} = -(|x_{uvi}| - (S_u \odot Q_i + Q_i \odot S_v + S_u \odot S_v))(S_u + S_v) + \lambda_I Q_i \quad (3.18)$$

The three aforementioned approaches, *PPMF, CDTF, PITF*, do not explicitly take into account the formulation of rating difference in their objective function. This could be a drawback since the current assumption on the structure of rating difference used in each approach (Equation 3.2, Equation 3.7 and Equation 3.13) may not well approximate the actual form of rating difference in Equation 3.1. Since rating differences are essentially constructed from ratings, we introduce another direction in the following section, in which $\hat{x}_{uvi}$ is estimated by applying Equation 3.1 on the estimation of the unseen ratings.

3.2.2 Factorizing Ratings

A straightforward approach to estimate unseen ratings is to fit a model to observed ratings, which can be done by applying matrix completion techniques such as matrix factorization on the rating matrix. This approach, *PMF*, assumes that fitting ratings will lead to fitting rating differences. This does not always hold, however, since fitting ratings optimizes a different objective function, i.e., minimizing residual error in predicted ratings. For instance, suppose that the true ratings are $r_{ui} = 4$ and $r_{vi} = 3$, which implies the rating difference

$x_{uvi} = 1$. If the rating prediction model has an error of 0.1 for each rating, we may end up with an estimation of $\hat{r}_{ui} = 4.1$ and $\hat{r}_{vi} = 3.1$ that still preserves the rating difference, or with an alternative estimation $\hat{r}_{ui} = 4.1$ and $\hat{r}_{vi} = 2.9$ that enlarges the rating difference.

We propose another approach, *DPMF*, that estimates unseen ratings by fitting a model to the observed rating differences, instead of observed ratings like *PMF*. In contrast to the rating difference-based methods in Section 3.2.1, *DPMF* explicitly integrates the structure of rating differences (Equation 3.1) in its objective function, giving its more advantages in accurately and efficiently capturing the observed rating differences. In the following, we first introduce the *PMF* model, followed by *DPMF*.

Rating Fitting. One way to predict $\hat{x}_{uvi}$ is to first predict $\hat{r}_{ui}$ and $\hat{r}_{vi}$, and subsequently taking their difference. As a representative of this approach, we employ the *Probabilistic Matrix Factorization* or *PMF* [76]. The set of ratings R can be represented as a matrix of size $|\mathcal{U}| \times |\mathcal{I}|$, where each element corresponds to a rating r_{ui}. This matrix is incomplete, and the goal is to fill up the missing entries with predicted $\hat{r}_{ui}$. The approximation uses two rank-K matrices $S \in \mathbb{R}^{K \times |\mathcal{U}|}$ and $Q \in \mathbb{R}^{K \times \mathcal{I}}$.

Let S_u be a column vector in S for consumer u. Let Q_i be a column vector in Q for item i. *PMF* places zero-mean spherical Gaussian priors on S_u and Q_i (with standard deviations $\varphi_{\mathcal{U}}$ and $\varphi_{\mathcal{I}}$) to control the complexity of the parameters, i.e., $S_u \sim \mathcal{N}(\mathbf{0}, \varphi_U^2 \mathbf{I})$ and $Q_i \sim \mathcal{N}(\mathbf{0}, \varphi_I^2 \mathbf{I})$. The plate diagram of PMF is shown in Figure 3.1(c). It shows how ratings are generated by the parameters S_u and Q_i. Each $\hat{r}_{ui}$ is assumed to be drawn from a Gaussian distribution centered at ${S_u}^T Q_i$ with variance γ^2 (Equation 3.19).

$$\hat{r}_{ui} \sim \mathcal{N}({S_u}^T Q_i, \gamma^2) \tag{3.19}$$

Parameter estimation is by maximizing the log-posterior distribution over item and consumer vectors with hyper-parameters, equivalent to minimizing

the sum of squared-errors function in Equation 3.20. $\mathbb{I}_R(u,i)$ is an indicator function of whether u has rated i. Equation 3.20 contains two components. The first summand is the fitting constraint, while the latter constitute the regularization. The fitting constraint keeps the model parameters fit to the training data. The regularizers avoid overfitting, making the model generalize better [37]. λ_U, λ_I are the regularization parameters.

$$E = \frac{1}{2}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbb{I}_R(u,i)(r_{ui} - {S_u}^T Q_i)^2 + \frac{\lambda_U}{2}\sum_{u\in\mathcal{U}}||S_u||^2 + \frac{\lambda_I}{2}\sum_{i\in\mathcal{I}}||Q_i||^2 \tag{3.20}$$

The estimation is done using gradient descent [76], with the following gradients. Once the parameters are learned, we then predict each $\hat{x}_{uvi}$ as ${S_u}^T Q_i - {S_v}^T Q_i$.

$$\frac{\partial E}{\partial S_u} = -(r_{ui} - {S_u}^T Q_i)Q_i + \lambda_U S_u \tag{3.21}$$

$$\frac{\partial E}{\partial Q_i} = -(r_{ui} - {S_u}^T Q_i)S_u + \lambda_I Q_i \tag{3.22}$$

Rating Difference Fitting. As discussed above, fitting ratings is an indirect way of predicting rating differences. Here we propose a new factorization model that meets our learning objective more directly, which we call *Differential Probabilistic Matrix Factorization* or *DPMF*. The plate diagram is shown in Figure 3.1(d). In this approach, we will still associate each consumer u with a latent vector S_u, and each item i with Q_i. The key distinction is that we consider ratings to be latent, and fit the rating difference x_{uvi} directly. In other words, $\hat{x}_{uvi}$ is a draw from the following Normal distribution (Equation 3.23).

$$\hat{x}_{uvi} \sim \mathcal{N}({S_u}^T Q_i - {S_v}^T Q_i, \gamma^2) \tag{3.23}$$

The objective function of *DPMF* in Equation 3.24 shows that we fit the prediction $\hat{x}_{uvi} = {S_u}^T Q_i - {S_v}^T Q_i$ to the observation $x_{uvi} = r_{ui} - r_{vi}$.

$$E = \frac{1}{2} \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}, v \neq u} \sum_{i \in \mathcal{I}} \mathbb{I}_R(u, v, i)((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))^2 + \frac{\lambda_U}{2} \sum_{u \in \mathcal{U}} ||S_u||^2 + \frac{\lambda_I}{2} \sum_{i \in \mathcal{I}} ||Q_i||^2 \tag{3.24}$$

Estimation by gradient descent uses the gradients below.

$$\frac{\partial E}{\partial S_u} = -((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))Q_i + \lambda_U S_u \tag{3.25}$$

$$\frac{\partial E}{\partial S_v} = ((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))Q_i + \lambda_U S_v \tag{3.26}$$

$$\frac{\partial E}{\partial Q_i} = -((r_{ui} - r_{vi}) - ({S_u}^T Q_i - {S_v}^T Q_i))(S_u - S_v) + \lambda_I Q_i \tag{3.27}$$

Note that both the approaches, *PMF* and *DPMF*, utilize the same number of parameters, but trained on different types of data, i.e., r_{ui} and x_{uvi} respectively, with different objective functions.

3.3 Experiments

In this section, we first examine the performance of different factorization methods for rating difference predictions under different epochs and different number of latent factors. Moreover, we show an application of rating differences to facilitate a special form of social recommendation in Section 3.3.3.

3.3.1 Experimental Setup

Datasets. We conduct experiments on four real-life, publicly available rating datasets, namely: Ciao[1], Epinions[1], Flixster[1], and Movielens100K[2]. Flixster

[1] `http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm`
[1] `http://www.cs.ubc.ca/~jamalim/datasets/`
[2] `http://grouplens.org/datasets/movielens/`

Table 3.1: Statistics on preprocessed data sets

Dataset	Users	Items	Ratings	User pairs	Rating Differences
Ciao	3.9×10^2	7.4×10^3	3.4×10^4	3.3×10^3	9.1×10^4
Epinions	1.1×10^3	2.4×10^5	1.3×10^5	1.0×10^4	3.6×10^5
Flixster	-	-	-	-	-
- Flixster06	1.8×10^2	3.4×10^3	6.7×10^4	1.6×10^3	3.0×10^5
- Flixster07	1.8×10^2	3.6×10^4	3.9×10^3	1.6×10^3	1.0×10^5
- Flixster08	1.8×10^2	3.0×10^4	2.1×10^4	1.6×10^3	6.5×10^4
- Flixster09	1.8×10^2	$2,1 \times 10^4$	1.4×10^4	1.6×10^3	4.8×10^4
MovieLens100K	9.0×10^2	1.5×10^3	9.9×10^4	1.2×10^5	6.0×10^6

and MovieLens100K contain ratings on movies. Ciao and Epinions both contain ratings on various categories such as books, electronics, movies, etc. We deliberately do not split the ratings by category to see if the model can contextualize the ratings per item basis without this information. Ratings are normalized into a 5-point scale. In all cases, only *ratings* (and not other information) are used in learning.

We pre-process the raw data as follows. First, we retain only pairs of users who have co-rated at least 20 items. We call such user pairs *neighbors*. This is to ensure that there is sufficient data to learn the model parameters reasonably accurately. For each co-rated item, we derive x_{uvi} from $r_{ui} - r_{vi}$. In addition, since Flixster has timestamps, we decide to split the ratings into four annual subsets: 2006-2009, and retain only user pairs who exist in all four subsets. This is to see if the results will be consistent across subsets of the data. The data sizes are shown in Table 3.1. After pre-processing, all the data sets are still sizeable, with thousands of users/items, and tens to hundreds of thousands rating differences.

Training vs. Testing. For each data set, we create two sets of training/testing data, namely *rating difference* data set and *rating* data set.

The first corresponds to the evaluation of the model components in Section 3.2, where we work with rating difference triplets x_{uvi}'s. We apply five-fold cross-validation on each pair (u, v)'s observed data with ratio of 80/20 for

training/testing set to ensure the pairs have all rating difference instances in both training/testing set. Let's denote X_{train} and X_{test} as a combined training/testing sample of all pairs. Each experiment is run five times on each of the five folds. The final result is reported as the average over the 25 runs for each setting.

The second corresponds to the evaluation of the integrated model for rating prediction in Sections 3.2.2 and 3.3.3, where we work with user-item ratings r_{ui}'s. To form the corresponding training set for ratings R_{train} for each X_{train}, we "decompose" each x_{uvi} into the original r_{ui} and r_{vi}. Similarly, R_{test} is created from X_{test}. To prevent duplicates and to maintain the ratio of training vs. testing, if any rating r_{ui} appears in both training and testing sets, it will be allocated randomly to the training set with probability 0.8 and to the testing set with probability 0.2. Note that each rating exists only in training set or testing set, but not both. Since there are five folds for X_{train} and X_{test}, correspondingly there are five folds for R_{train} and R_{test}.

This experimental setup ensures the fairness for comparing all the methods. First, the training and testing sets are stratified into five independent folds that support the traditional cross validation approach. Second, the ratings that exist in both training and testing sets are distributed between them to guarantee that each user/item has instances in training and testing set. It is fair for all the methods since they have access to exactly the same information, and is more reflective of the utility of *DPMF* in rating differences prediction.

3.3.2 Experimental Results

We study the efficacy of different factorization methods outlined in Section 3.2 in deriving good rating difference predictions. For each dataset, we use the same five folds as before. All except *PMF* are trained on X_{train}, while *PMF* is trained on the corresponding R_{train}. Parameter settings are adopted from the original paper for PMF [76] (learning rate = 0.005, number of latent factors

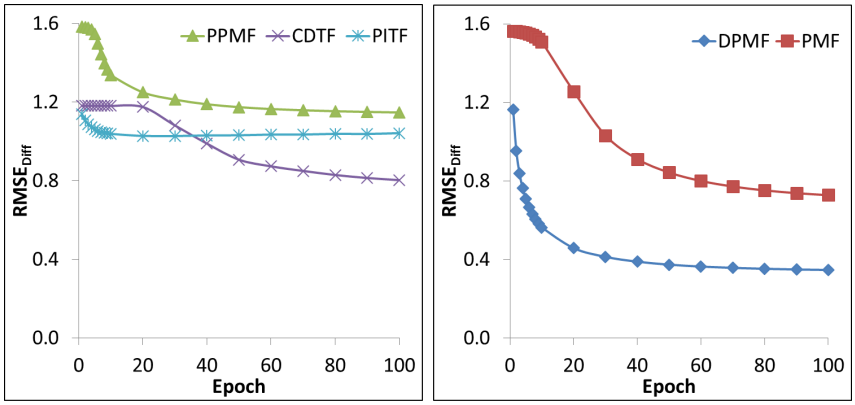


Figure 3.2: Comparison of Rating Difference Prediction Methods ($RMSE_{diff}$) on Ciao dataset

= 30, regularization coefficient = 0.002). All models are tested on the same X_{test}.

For every triplet x_{uvi} in the test set X_{test}, we derive a prediction $\hat{x}_{uvi}$ using each method, and compare the accuracy of their predictions in terms of root mean squared error commonly used in matrix factorization. $RMSE_{diff}$ is defined in Equation 3.28. We use absolute values because only magnitudinal error affects the contextual probability $\mathrm{P}(y|x_{uvi})$. Lower $RMSE_{diff}$ value indicates better performance.

$$RMSE_{diff} = \sum_{x_{uvi} \in X_{test}} \sqrt{\frac{(|\hat{x}_{uvi}| - |x_{uvi}|)^2}{|X_{test}|}} \tag{3.28}$$

Vary Epochs. In Figure 3.2, we plot the $RMSE_{diff}$ across epochs on Ciao dataset. One epoch corresponds to a full iteration over the whole training set. By 100 epochs, all the factorization methods have converged.

Comparing the three approaches that factorize rating differences in Figure 3.2(a), we observe that the tensor-based approaches *CDTF* and *PITF* perform better than the pairwise matrix factorization *PPMF*. We attribute this to the tensor factorization approach that ties together the latent vector for each user in different triplets.

Comparing the two approaches that factorize ratings in Figure 3.2(b), we observe that *DPMF* converges faster than *PMF*, and achieves a lower

Table 3.2: Performance of Rating Difference Prediction Methods ($RMSE_{diff}$) for 100 epochs and $k = 30$ (statistically significant best-performing entries are asteriated)

	PPMF	CDTF	PITF	PMF	DPMF	Mean
Ciao	1.15	0.8	1.04	0.73	0.35*	1.18
Epinions	0.91	0.82	0.82	0.62	0.32*	0.89
Flixster06	0.57	0.74	0.70	0.55	0.47*	0.81
Flixster07	0.68	0.73	0.71	0.45	0.38*	0.79
Flixster08	0.71	0.66	0.64	0.44	0.33*	0.73
Flixster09	0.72	0.65	0.61	0.40	0.26*	0.72
MovieLens100K	1.06	0.89	0.83	0.74	0.70*	0.89

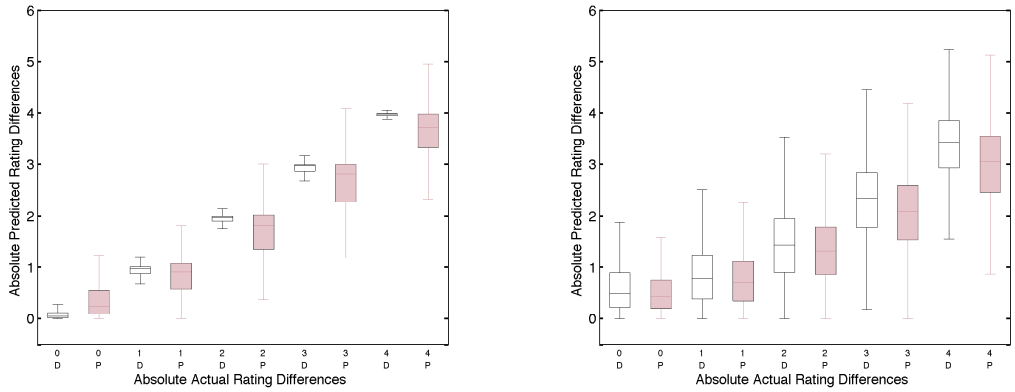
$RMSE_{diff}$ value. We attribute this to the approach of fitting the rating differences.

Comparing all the approaches across all datasets in Table 3.2, we observe that *DPMF* performs the best, followed by *PMF*. The tensor-based models *CDTF* and *PITF* have middling performance, better than *PPMF* but worse than *DPMF* and *PMF*. They do not directly reflect the generation process of x_{uvi}. As defined in Equation 3.1, x_{uvi} is modeled as the difference between r_{ui} and r_{vi}. *CDTF* models x_{uvi} as inner products of three vectors (Equation 3.7), which may find difficulties in capturing the correlations in rating differences. Although *PITF* explicitly addresses the underlying dyadic interactions (Equation 3.13), its additive form does not reflect differences between two ratings.

We also introduce a baseline *Mean*, which simply averages the absolute values of all training instances as the prediction. Table 3.2 shows that all the proposed models outperform this simple baseline.

We perform one-tailed paired samples t-test with 0.01 significance level on the $RMSE_{diff}$ values of *DPMF* and other other comparable methods over different epochs. The result confirms that the outperformance by *DPMF* is statistically significant.

To better understand why *DPMF* has better performance than *PMF* in predicting rating differences, we conduct a deeper investigation of the the two methods on two datasets: *Ciao* and $Movielens100K$ dataset. In Figure 3.3,



(a) Ciao (b) Movielens100K

Figure 3.3: Comparison of Predicted vs. Observed Rating Differences for DPMF (white) vs PMF (pink) on Ciao and Movielens100K.

the horizontal axes line up the observed rating differences, which range from 0 to 4. The maximum difference 4 is the difference between the lowest (1) and highest (5) ratings. The vertical axes show the range of predictions (median and inter-quartile) made by DPMF (white) and PMF (pink) respectively. First, we see that DPMF has much lower variances (narrow inter-quartile ranges) across all the bins, implying that its predictions are more precise. Second, the medians by DPMF are also closer to the actual observed rating differences than those by PMF. The degree of outpeformance varies across datasets. There is greater difference between DPMF and PMF's performances on *Ciao* than on $Movielens100K$.

One reason for DPMF's outperformance is given in the beginning of Section 3.2. DPMF has an error function that fits rating differences directly. In contrast, PMF seeks to fit ratings, which may not necessarily fit the rating differences as well, as the rating error may enlarge or narrow the rating differences. Another reason we posit here concerns the type of training instances. There are fewer rating instances than rating difference instances. Hence, it may cost PMF many more training epochs to learn effectively from the rating instances.

Vary Latent Factors. We conduct a separate experiment on *DPMF* for

Table 3.3: DPMF: Vary Latent Factors ($RMSE_{diff}$)

	Number of latent factors K				
Data set	10	20	30	40	50
Ciao	0.85	0.43	0.35	0.34	0.34
Epinions	0.71	0.44	0.32	0.30	0.29
Flixster06	0.72	0.57	0.47	0.39	0.34
Flixster07	0.60	0.45	0.38	0.35	0.35
Flixster08	0.56	0.40	0.32	0.31	0.30
Flixster09	0.53	0.32	0.27	0.25	0.25
MovieLens100K	0.89	0.79	0.69	0.58	0.46

different numbers of latent factors K. The $RMSE_{diff}$ at 100 epochs are shown in Table 3.3. It shows that by around $K = 30$, the errors have converged. There is no significant gain by running higher latent factors (which will make the learning algorithms slower). Subsequently, we will use *DPMF* in conjunction with *CAM* with the same parameter settings ($K = 30$, 100 epochs) .

The gradient descent learning algorithms are also efficient. For all methods, the parameters can be learned within 5 minute for each fold on the same Intel(R) Xeon(R) Processor E5-2667 2.90GHz machine.

3.3.3 Application: Finding the More Similar Neighbors

In this section we show an application of rating differences to facilitate a special form of social recommendation. For instance, a consumer may wish to recommend an item only to specific neighbors [14], who would be expected to have a similar response. Given u and one of her adopted/rated items i, the task is to rank two of her neighbors v and z, whose preferences on i are not observed, based on their predicted rating differences with u on i. Because we observe the actual rating by v and z respectively on i (which are held out), we can determine the ground truth based on the observed $|x_{uvi}|$ and $|x_{uzi}|$ (smaller absolute difference is higher similarity).

We create training and test sets for this application from the data sets above as follows. For every consumer u, we randomly select an item i among

Table 3.4: Ranking application with Kendall's Tau (statistically significant best-performing entries are asteriated)

Dataset	*DPMF*	Shared Preference	
		Cosine	Pearson
Ciao	0.122*	0.051	-0.028
Epinions	0.070*	0.059	0.028
Flixster06	0.223*	0.115	0.035
Flixster07	0.149*	0.039	0.021
Flixster08	0.222*	0.080	0.031
Flixster09	0.240*	0.117	0.018
MovieLens100K	0.119*	0.079	0.021

the set of items that u rated. Two neighbors v and z are then randomly select with different real ratings to i, r_{vi} and r_{zi}. Both ratings are held out from the training set. Therefore, the test set consists of several testing tuples in form of $(u, i, v, z, r_{vi}, r_{zi})$. We sample at least sixty such folds.

For every testing tuple, v and z are ranked based on the similarity scores, w_{uvi} and w_{uzi},which are computed using Cosine or Pearson's, or the predicted rating differences of *DPMF*. The accuracy of ranking is measured by using the Kendall's Tau coefficient [53]. A pair (u, v) and (u, z) are said to be concordant if the held out $|x_{uvi}| < |x_{uzi}|$ and the similarity $w_{uv} > w_{uz}$ for Cosine and Pearson, or $|\hat{x}_{uvi}| < |\hat{x}_{uzi}|$ for *DPMF*, are consistent (i.e., $|x_{uvi}| < |x_{uzi}|$ and $w_{uv} > w_{uz}$ or $|x_{uvi}| < |x_{uzi}|$ and $|\hat{x}_{uvi}| < |\hat{x}_{uzi}|$). Otherwise, that pair are discordant. Let us denote C and $\bar{C}$ as the number of concordant and discordant pairs. The overall Kendall's Tau coefficient is computed as $\tau = \frac{C-\bar{C}}{C+\bar{C}}$. The range of τ is therefore within $[-1, 1]$. Higher value of τ indicates better performance in ranking. A random ranking would result in $\tau = 0$.

Table 3.4 shows the results of various methods across datasets. Each cell's value is an average over ten different runs. Since *DPMF* is shown to be effective in predicting rating differences in Section 3.3.2, it is reasonable for *DPMF* to have the highest τ coefficient. The numbers also show that Cosine and Pearson have lower τ coefficients, and their lower performance is due to using the same similarity across all items, rather than specific to each item.

3.4 Discussion

In this chapter we discuss the problem of predicting rating differences. Our motivation is that differences in ratings signal the agreement or disagreement in preferences between consumers on the product basis. We examine various factorization methods on extensive experiments to find the best performer for the problem, *DPMF*. We also show the effectiveness of *DPMF* through an application of finding more relevant neighborhoods.

As described in Section 3.2.1, *DPMF* only captures how two consumers have different ratings on a single product. A rating difference of 1 may indicate a disagreement if the two always give similar ratings, or an agreement if the two always rate differently. We posit that a precise measure of agreement in preferences needs to take all the rating differences between the pair into account, instead of only one instance. As each consumer pair may have their own range of agreement, in the next chapter, we present a probabilistic approach to derive the agreement measure specific to their data.

Chapter 4

Modeling Contextual Agreement in Preferences

4.1 Overview

As mentioned in Chapter 2.1, the *individual preference* approach relies heavily on individual's data, posing challenges in learning a valid estimation of the utility function for consumers with a few feedback. One approach to tackle those "cold-start" consumers is to enrich their information from others. We call this approach *shared preference*.

The gist of *shared preference* is to determine the consumers suitable for sharing and how to share the information. Intuitively, consumers with similar tastes would give similar feedback. This brings up to the notion of measuring similarity in preferences between pairs of consumers from their responses. There are two common sources often considered to measure similarity, namely ratings or existing relationship (e.g., friends or follower-followee) in social network. For the latter source, each connection is seen as inducing sharing of preferences between the two consumers [71, 70]. Most works on utilizing the social network structure are found in latent factor-based models. The general idea across those works is to use the connections as prior belief to force the

latent parameters between two social friends to be as similar as possible.

As relationship in social network is not often given, there are similarity measures defined based on the rating data. Intuitively, the agreement in preferences between two consumers is reflected through their ratings on co-rated products. By representing each consumer by a rating vector across products, we can apply similarity measures in Information Retrieval which measure the similarity between two vectors, such as Pearson's correlation coefficient [84], and vector space or Cosine similarity [20]. Denote w_{uv} is the similarity between a pair of users u and v. The higher w_{uv} is, the more u and v agree in their preferences. Given that $\mathbf{r}_u$ and $\mathbf{r}_v$ represent vectors of ratings, $\{r_{ui}\}$ by u and $\{r_{vi}\}$ by v, on a set of items $\{i\}$, Pearson is determined as in Equation 4.1 (where $\bar{r}_u$ and $\bar{r}_v$ are average ratings), and Cosine as in Equation 4.2. For item-based CF [89, 64], the similarity is between a pair of items.

$$w_{uv}^{pearson} = \frac{\sum_i (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_i (r_{vi} - \bar{r}_v)^2}} \tag{4.1}$$

$$w_{uv}^{cosine} = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u|| \times ||\mathbf{r}_v||} \tag{4.2}$$

Both measures in Equation 4.1 and 4.2 implicitly assume that the similarity between two consumers applies equally to *all* products under consideration. Realistically, consumers have diverse preferences. While two consumers agree in their preferences in some products, they may disagree in other products. A single similarity score such as $w_{uv}^{pearson}$ (or w_{uv}^{cosine}) cannot reflect such diversity of preferences across various products. For example, a high value of $w_{uv}^{pearson}$ would overestimate the similarity of the pair on products they usually do not agree with each other.

To capture the diversity of preferences between two consumers across products, we propose to explicitly incorporate products as an additional dimension in measuring similarity. Essentially, we look for a similarity measure which

returns w_{uvi}, or the similarity score indicating how u and v agree in their preferences toward product i. Comparing to w_{uv} which indicates a global similarity, w_{uvi} is more "contextualized" since it can return different value regarding to the product in consideration. For item-based CF, the similarity between an item pair can similarly be contextualized regarding to each consumer. To contrast with single-valued similarity measures, we call w_{uvi} a measure of *contextual agreement* of preferences. Note that here we slightly abuse the notion of "context" which often indicates additional information such as time, location, etc., in the literature. By "context" we refer to each specific product (or consumer) involved in the similarity measure.

Based on the same observations on observed rating values in Section 3.1, we study the *contextual agreement* of preferences on rating differences. Note that our focus in this chapter is to measure how two consumers agree in their preferences on a specific product, instead of how their ratings on that product differ as addressed in the previous chapter. Intuitively, rating difference itself signals some degree of agreement in preferences. However, the degree of agreement is more precisely measured with a full consideration over the pair's observed range of rating differences. In this chapter, we propose a probabilistic generated model, called *Contextual Agreement Model* or *CAM*, which takes rating differences of a consumer pair as inputs and returns product-specific values indicating their contextual agreement of preferences.

4.1.1 Problem Statement

We extend the problem statement in Section 3.1.1 as follow.

Notations. Each triplet $\langle u, v, i \rangle$ of two consumers u and v on one product i is associated with two quantities (modeled as random variables): x_{uvi} and y_{uvi}, which are essential to our probabilistic modeling.

The variable $x_{uvi} \in \mathbb{R}$ represents the indicator of agreement between u and v on i, some of which are observed in the data. The closer is x_{uvi} to 0,

the more likely it is that u and v agree on i. If $x_{uvi} \ll 0$ or $x_{uvi} \gg 0$, then disagreement is more likely. x_{uvi} can be expressed as a function of ratings, i.e., $x_{uvi} = \mathcal{F}(r_{ui}, r_{vi})$. In this chapter, we simply use the *rating difference* between two consumers on the same item, as shown in Equation 4.3.

$$x_{uvi} = r_{ui} - r_{vi} \tag{4.3}$$

The second variable $y_{uvi} \in \mathcal{Y} = \{0, 1\}$ is binary. $y_{uvi} = 1$ represents the event of agreement between u and v on their preference for i. $y_{uvi} = 0$ is the event of disagreement. These events are latent, and are to be estimated from the observed x_{uvi}'s.

Problem Formulation. Given rating data R, and the above x_{uvi} definition, we seek to estimate the probability $\mathrm{P}(y_{uvi}|x_{uvi})$ for all triplets. Not all x_{uvi}'s can be observed. x_{uvi} is not observed if either $r_{ui} \notin R$ or $r_{vi} \notin R$. This gives rise to two sub-problems. The first is how to estimate $\mathrm{P}(y_{uvi}|x_{uvi})$ given the observed x_{uvi} values. The second sub-problem is how to predict the unobserved $\hat{x}_{uvi}$ values.

For the first sub-problem, we propose the probabilistic *CAM* model in Section 4.2. Since y_{uvi} is latent, we turn to generative modeling, by representing x_{uvi} as a random variable, whose generative process is related to y_{uvi}. Our approach is thus to model the joint probability $\mathrm{P}(y_{uvi}, x_{uvi})$. The conditional probability $\mathrm{P}(y_{uvi}|x_{uvi})$ can afterwards be estimated from the joint probabilities as follows:

$$\mathrm{P}(y_{uvi}|x_{uvi}) = \frac{\mathrm{P}(y_{uvi}, x_{uvi})}{\sum_{y'_{uvi} \in \mathcal{Y}} \mathrm{P}(y'_{uvi}, x_{uvi})} \tag{4.4}$$

The second sub-problem is how to predict the unseen $\hat{x}_{uvi}$. This is also the problem addressed in Chapter 3. Recall that our approach is to model the generation of x_{uvi} based on user- or item-specific parameters to generate/predict unseen $\hat{x}_{uvi}$ through matrix or tensor factorization in Section 3.2. We use *DPMF* in this chapter for its best performance compared to other models.

4.2 Methodology

In this section, we describe the generative model for *CAM*, outline the monotonicity property of its "decision function", and develop an algorithm to learn its parameters.

4.2.1 Generative Model

Given the observed x_{uvi}'s, we estimate the probability distribution of contextual agreement $\mathrm{P}(y_{uvi}|x_{uvi})$. When the context is clear, we simplify the notations for y_{uvi} and x_{uvi} to y and x respectively. Because y is latent, we estimate the conditional probability $\mathrm{P}(y|x)$ from the joint probability $\mathrm{P}(y, x)$. In a generative modeling framework, we decompose $\mathrm{P}(y, x)$ into $\mathrm{P}(x|y)\mathrm{P}(y)$. $\mathrm{P}(y)$ corresponds to the prior probability of agreement between u and v on i. $\mathrm{P}(x|y)$ is the likelihood that x has been generated from y.

The prior of agreement $\mathrm{P}(y)$ is the base level of agreement between u and v before seeing the item i. Given that there are two probable events, i.e., agreement ($y = 1$) and disagreement ($y = 0$), we model this as a Bernoulli process with a parameter α. In other words, the prior of agreement is $\mathrm{P}(y = 1) = \alpha$, and of disagreement is $\mathrm{P}(y = 0) = 1 - \alpha$.

In the event of agreement ($y = 1$), x is generated according to $\mathrm{P}(x|y = 1)$. As x is real-valued, and we expect that its values will cluster together in the event of agreement, we model its generation as a Gaussian, with a mean μ_1 and variance σ_1^2. As mentioned in Section 4.1, the closer x_{uvi} is to 0, the more likely it is that u and v agree on i. Therefore, we make a simplifying step, and set $\mu_1 = 0$. We learn σ_1 from data. The blue curve in Figure 4.1(a) illustrates the probability density function (p.d.f.) of $\mathrm{P}(x|y = 1)$, which is a Normal distribution centered at $\mu_1 = 0$ (in this example, $\sigma_1 = 0.9$).

In the event of disagreement ($y = 0$), x is generated according to $\mathrm{P}(x|y = 0)$. Since $x \gg 0$ or $x \ll 0$ indicates disagreement, the mean of this Gaussian

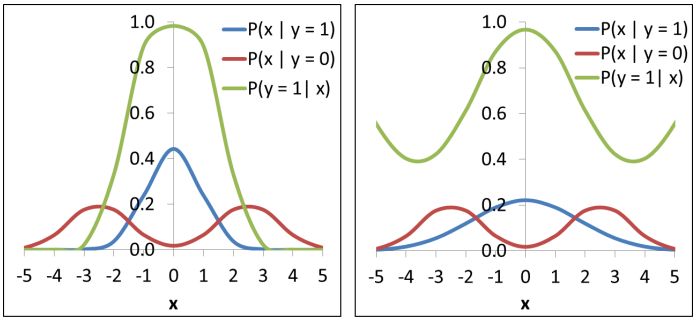


Figure 4.1: Distributions of $\mathrm{P}(x|y)$ and $\mathrm{P}(y|x)$

should be away from 0. Due to the symmetric property $x_{uvi} = -x_{vui}$, a reasonable model is a bimodal distribution, such as an equally-weighted mixture of two Gaussians with positive mean at μ_0 and negative mean $-\mu_0$, and a variance of σ_0^2. The red curve on Figure 4.1(a) illustrates the bimodal p.d.f. of $\mathrm{P}(x|y=0)$ (in this example, $\mu_0 = 2.5$, $\sigma_0 = 1$).

$\mathrm{P}(y|x)$ can therefore be expressed in terms of these components as shown in Equation 4.5. The green curve on Figure 4.1(a) illustrates the "decision function" or the p.d.f. of $\mathrm{P}(y=1|x)$, estimated from the respective prior $\mathrm{P}(y)$ and likelihood $\mathrm{P}(x|y)$. As expected, $\mathrm{P}(y=1|x)$ is highest when $x \approx 0$. As x moves away from 0, the probability of agreement decreases, which fits the modeling objective.

$$\mathrm{P}(y|x) = \frac{\mathrm{P}(x|y)\mathrm{P}(y)}{\sum_{y' \in \mathcal{Y}} \mathrm{P}(x|y')\mathrm{P}(y')} \tag{4.5}$$

Generative Process. We now describe the full generative process for a set of observed triplets $X = \{x\}$.

For every triplet $x \in X$:

1. Draw an outcome for $y \in \{0, 1\}$:

$$y \sim Bernoulli(\alpha)$$

2. Draw an outcome for $x \in \mathbb{R}$:

(a) In the event of agreement, i.e., $y = 1$:

$$x \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

(b) Else, in the event of disagreement, i.e., $y = 0$:

$$x \sim \frac{1}{2}\mathcal{N}(\mu_0, \sigma_0^2) + \frac{1}{2}\mathcal{N}(-\mu_0, \sigma_0^2)$$

Based on this generative process, the distribution of x can be expressed as a mixture of three Gaussians with weights α, $\frac{1-\alpha}{2}$, and $\frac{1-\alpha}{2}$ respectively, as shown in Equation 4.6.

$$x \sim \alpha\mathcal{N}(\mu_1, \sigma_1^2) + \frac{1-\alpha}{2}\mathcal{N}(\mu_0, \sigma_0^2) + \frac{1-\alpha}{2}\mathcal{N}(-\mu_0, \sigma_0^2) \tag{4.6}$$

Parameters. For the above generative process, the set of parameters can be encapsulated by $\theta = \langle \alpha, \mu_1, \sigma_1, \mu_0, \sigma_0 \rangle$. The question arises whether there is a unique θ for every triplet $\langle u, v, i \rangle$. Because θ is a distributional parameter, it is not feasible to estimate θ from a single observation of x. Another approach is to tie together the parameters of a group of triplets. In this paper, we will experiment with two approaches. First is the *Global* parameter, where θ is shared by all triplets. Second is the *Local* parameter, where there is a specific θ_{uv} for each pair of consumers u and v that applies to all items. The distinction between these two approaches can be seen clearly in the plate diagrams in Figure 4.2. For clarity, we draw α separately to show that y_{uvi} only depends on α, although $\alpha \in \theta$. In both cases, x_{uvi} is shaded, because they form the observations. For *Local*, θ_{uv} is within the plate of each pair of consumers. For *Global*, θ is outside.

Notation. For readability, we use the following notations in the remainder of this section.

- $\phi(x) = \exp\left\{-\frac{1}{2}x^2\right\}$

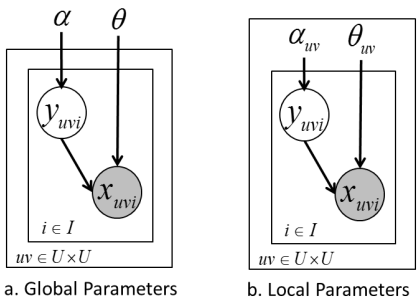


Figure 4.2: Global vs. Local Parameters

- $\phi_1 = \phi\left(\frac{x}{\sigma_1}\right) = \exp\left\{-\frac{x^2}{2\sigma_1^2}\right\}$
- $\phi_0^+ = \phi\left(\frac{x+\mu_0}{\sigma_0}\right) = \exp\left\{-\frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}$
- $\phi_0^- = \phi\left(\frac{x-\mu_0}{\sigma_0}\right) = \exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right\}$

4.2.2 Monotonicity Property

We would like to model $\mathrm{P}(y = 1|x)$ that increases as $x \to 0$, and decreases as $x \to \infty$ or $x \to -\infty$. We refer to this as the monotonicity property of the conditional probability of agreement. This monotonicity property does not always hold for all parameter settings. There are errant parameter settings that may cause this property to be violated. As an example, in Figure 4.1(b), we show a case where $\mathrm{P}(y = 1|x)$ (the green curve) initially decreases as x goes away from zero, but as x continues moving away, it starts to increase again. This is counter intuitive, as it suggests that the probability of agreement is very high even as $x \to \infty$.

To enforce the monotonicity property, we propose introducing some constraint to the parameters of the Gaussian mixtures. By expanding Equation 4.5 according to the generative process, we can express the p.d.f. of $\mathrm{P}(y = 1|x)$ as in Equation 4.7.

$$\mathcal{G}(x) = \frac{\alpha\mathcal{N}(x; 0, \sigma_1^2)}{\alpha\mathcal{N}(x; 0, \sigma_1^2) + \frac{1-\alpha}{2}\mathcal{N}(x; \mu_0, \sigma_0^2) + \frac{1-\alpha}{2}\mathcal{N}(x; -\mu_0, \sigma_0^2)} \tag{4.7}$$

Here, $\mathcal{N}(x; \mu, \sigma^2)$ denotes the p.d.f. of Normal distribution with two parameters u and σ.

Because the p.d.f $\mathcal{G}(x)$ is continuous and differentiable, one way to ensure monotonicity is to constrain the gradient of $\mathcal{G}(x)$ to be negative for $x > 0$, as shown in Equation 4.8. Note that due to the symmetric property of the Gaussian mixtures, it is sufficient to enforce this monotonicity for $x > 0$, as the other case $x < 0$ is met simultaneously.

$$\frac{\partial \mathcal{G}(x)}{\partial x} < 0, \text{for all } x > 0 \tag{4.8}$$

By $A(x)$ and $B(x)$ we denote the numerator and denominator of the above equation, i.e., $\mathcal{G}(x) = A(x) \div B(x)$. Both $A(x)$ and $B(x)$ are shown in Equation 4.9.

$$\begin{aligned} A(x) &= \alpha \mathcal{N}(x; 0, \sigma_1) = \alpha \frac{1}{\sigma_1} \frac{1}{\sqrt{2\pi}} \phi_1 \\ B(x) &= \alpha \mathcal{N}(x; 0, \sigma_1) + 0.5(1-\alpha)(\mathcal{N}(x; \mu_0, \sigma_0) + \mathcal{N}(x; -\mu_0, \sigma_0)) \\ &= \alpha \frac{1}{\sigma_1} \frac{1}{\sqrt{2\pi}} \phi_1 + 0.5(1-\alpha) \frac{1}{\sigma_0} \frac{1}{\sqrt{2\pi}} (\phi_0^- + \phi_0^+) \end{aligned} \tag{4.9}$$

The derivative of $\mathcal{G}(x)$ thus can be expressed in terms of $A(x)$ and $B(x)$.

$$\frac{\partial \mathcal{G}(x)}{\partial x} = \frac{A'(x)B(x) - B'(x)A(x)}{(B(x))^2} < 0 \tag{4.10}$$

The first derivative of $A(x)$ w.r.t. x, or $A'(x)$, is as follows.

$$A'(x) = \alpha \frac{-x}{\sigma_1^3} \frac{1}{\sqrt{2\pi}} \phi_1 \tag{4.11}$$

The first derivative of $B(x)$ w.r.t. x, or $B'(x)$ is as follows.

$$B'(x) = \frac{-\alpha x}{\sigma_1^3}\frac{1}{\sqrt{2\pi}}\phi_1 - \frac{1-\alpha}{2\sigma_0^3}\frac{1}{\sqrt{2\pi}}(\phi_0^-(x-\mu_0) + \phi_0^+(x+\mu_0)) \tag{4.12}$$

For Equation 4.10 to hold, the condition in Equation 4.13 must be satisfied.

$$A'(x)B(x) - B'(x)A(x) < 0 \Leftrightarrow A'(x)B(x) < B'(x)A(x) \tag{4.13}$$

Substituting $A'(x)$ and $B'(x)$ into Equation 4.13, we have the following inequality.

$$\begin{aligned} &- \alpha\frac{x}{\sigma_1^3}\frac{1}{2\pi}\phi_1\left(\frac{\alpha}{\sigma_1}\phi_1 + \frac{1-\alpha}{2\sigma_0}\left(\phi_0^- + \phi_0^+\right)\right) \\ &< \alpha\frac{1}{\sigma_1}\frac{1}{2\pi}\phi_1(\frac{-\alpha x}{\sigma_1^3}\phi_1 - \frac{1-\alpha}{2\sigma_0^3}(\phi_0^-(x-\mu_0) + \phi_0^+(x+\mu_0))) \end{aligned} \tag{4.14}$$

Cancelling α, $\frac{1}{\sigma_1}$, $\frac{1}{2\pi}$ and ϕ_1 from both sides of Equation 4.14, we have:

$$\frac{x}{\sigma_1^2}\left(\frac{\alpha}{\sigma_1}\phi_1 + \frac{1-\alpha}{2\sigma_0}\left(\phi_0^- + \phi_0^+\right)\right) > \frac{\alpha x}{\sigma_1^3}\phi_1 + \frac{1-\alpha}{2\sigma_0^3}\left(\phi_0^-(x-\mu_0) + \phi_0^+(x+\mu_0)\right) \tag{4.15}$$

Subtracting the term $\alpha\frac{x}{\sigma_1^3}\phi_1$ from both sides of Equation 4.15, we get:

$$\frac{1-\alpha}{2\sigma_0}\frac{x}{\sigma_1^2}\left(\phi_0^- + \phi_0^+\right) > \frac{1-\alpha}{2\sigma_0^3}\left(\phi_0^-(x-\mu_0) + \phi_0^+(x+\mu_0)\right) \tag{4.16}$$

Cancelling 0.5, $1-\alpha$ and $\frac{1}{\sigma_0}$ from both sides of Equation 4.16, we obtain:

$$\frac{x}{\sigma_1^2}\left(\phi_0^- + \phi_0^+\right) > \frac{1}{\sigma_0^2}(\phi_0^-(x-\mu_0) + \phi_0^+(x+\mu_0)) \tag{4.17}$$

Reorganizing Equation 4.17 by moving all terms to one side, we achieve

$$\begin{aligned}
&\phi_0^-\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \phi_0^+\left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) > 0 \\
\Leftrightarrow &\exp\left\{-\frac{(x-\mu_0)^2}{2\sigma_0^2} + \frac{(x+\mu_0)^2}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_0^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) > 0 \\
\Leftrightarrow &\exp\left\{\frac{4x\mu_0}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) > 0
\end{aligned} \tag{4.18}$$

The final inequality in Equation 4.18 still contains the variable x. We need to reduce it to an inequality involving only parameters. We discover a simple constraint that meets the objective.

Proposition 1. *The constraint $\sigma_1 < \sigma_0$ ensures that Equation 4.18 always holds for any $x > 0$.*

Proof. Let us first consider the first additive term in the LHS of the last inequality in Equation 4.18, i.e., $\exp\{\frac{4x\mu_0}{2\sigma_0^2}\}(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2})$. Because x, μ_0, and σ_0 are all positive, we have $\frac{4x\mu_0}{2\sigma_0^2} > 0$. In turn, we have $\exp\{\frac{4x\mu_0}{2\sigma_0^2}\} > 1$. Because $\sigma_1 < \sigma_0$, we also have $(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}) > 0$. We can therefore take Step 1 in Equation 4.19.

$$\begin{aligned}
&\exp\left\{\frac{4x\mu_0}{2\sigma_0^2}\right\}\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) && (4.19)\\
\geq &\left(\frac{x}{\sigma_1^2} - \frac{x-\mu_0}{\sigma_0^2}\right) + \left(\frac{x}{\sigma_1^2} - \frac{x+\mu_0}{\sigma_0^2}\right) && \text{(Step 1)}\\
= &2x\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_0^2}\right) && \text{(Step 2)}\\
> &0 && \text{(Step 3)}
\end{aligned}$$

From Step 1, we can go to Step 2 by a simple addition of the terms. Finally, because $x > 0$, and $\sigma_1 < \sigma_0$, we have $2x(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_0}) > 0$ in Step 3, which concludes the proof. □

We have shown that with the constraint of $\sigma_1 < \sigma_0$, Equation 4.8 holds, guaranteeing the monotonicity property for $x > 0$ (and simultaneously for $x < 0$). This constraint $\sigma_1 < \sigma_0$ is also intuitive, as when two consumers are agreeing their rating difference is likely to be small and not vary as widely as when they are disagreeing.

4.2.3 Parameter Estimation

We seek to learn the parameters θ that best "describe" the observed data $X = \{x\}$. Because every x is assumed to have been generated independently in the generative process, the likelihood can be expressed as the joint probability shown in Equation 4.20.

$$\mathrm{P}(X|\theta) = \prod_{x \in X} \mathrm{P}(x|\theta) \tag{4.20}$$

The strategy employed in this paper is to find the parameters that maximize the likelihood of observing X. Due to the presence of constraints, the objective is to also find θ that meets the constraints, as shown in Equation 4.21. The first constraint ensures the mixture weights of the Gaussians sum to 1, by setting the mixture weights to $\alpha_1 = \alpha$ and $\alpha_0 = 1 - \alpha$ respectively. The second constraint ensures the monotonicity of $\mathrm{P}(y = 1|x)$ by setting $\sigma_1 < \sigma_0$.

$$\begin{aligned} &\arg\max_{\theta} \mathrm{P}(X|\theta), \\ &\text{subject to: } \alpha_0 + \alpha_1 = 1, \text{and } \sigma_1 < \sigma_0 \end{aligned} \tag{4.21}$$

To maximize the likelihood, we can equivalently maximize the log-likelihood. As it is a constrained optimization problem, we employ the use of Lagrangian

multipliers [19] to enforce the constraint. In Equation 4.22, we show the updated log-likelihood function $\mathcal{L}$. Both λ_α and λ_σ are Lagrangian multipliers. We introduce a slack variable s^2, whose positive value ensures that $\sigma_1 < \sigma_0$.

$$\mathcal{L} = \sum_{x \in X} \ln \mathrm{P}(x|\theta) + \lambda_\alpha(\alpha_1 + \alpha_0 - 1) + \lambda_\sigma(\sigma_0 - \sigma_1 - s^2) \tag{4.22}$$

To learn the parameters that maximize the log-likelihood function $\mathcal{L}$, we turn to the Expectation Maximization (EM) algorithm [16]. To outline the E-step and M-step, we first find the derivatives of $\mathcal{L}$, with respect to each individual parameter in θ. These derivations are shown in the following.

Derivation of $\mathcal{L}$ w.r.t to μ_0

Equation 4.23 shows the differentiation of Equation 4.22 with respect to μ_0.

$$\begin{aligned}
\frac{\partial}{\partial \mu_0}\mathcal{L} &= \sum_{x \in X} \frac{\partial}{\partial \mu_0} ln \mathrm{P}(x|\theta) \\
&= \sum_{x \in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{\partial}{\partial \mu_0} \mathrm{P}(x|\theta) \\
&= \sum_{x \in X} \frac{1}{\mathrm{P}(x|\theta)} (1-\alpha) \frac{1}{2\sigma_0\sqrt{2\pi}} \left(\phi_0^+ \frac{-(x+\mu_0)}{\sigma_0^2} + \phi_0^- \frac{(x-\mu_0)}{\sigma_0^2} \right) \\
&= \sum_{x \in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{1-\alpha}{2\sigma_0^2} \big(x(\mathcal{N}(x| -\mu_0, \sigma_0^2) - \mathcal{N}(x|\mu_0, \sigma_0^2)) \\
&\qquad\qquad -\mu_0(\mathcal{N}(x| -\mu_0, \sigma_0^2) + \mathcal{N}(x|\mu_0, \sigma_0^2))\big)
\end{aligned} \tag{4.23}$$

In the **E-step**, we compute $C = \sum_{x \in X} c(x)$, where $c(x)$ is defined in Equation 4.24.

$$c(x) = \frac{1}{\mathrm{P}(x|\theta)} \frac{1-\alpha}{2} (\mathcal{N}(x| -\mu_0, \sigma_0^2) + \mathcal{N}(x|\mu_0, \sigma_0^2)) \tag{4.24}$$

Set Equation 4.23 to zero to solve for μ_0. In the **M-step**, we use C to

update μ_0, as shown in Equation 4.25.

$$\begin{aligned}
\sum_{x \in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{1-\alpha}{2} (\mathcal{N}(x| - \mu_0, \sigma_0^2) + \mathcal{N}(x|\mu_0, \sigma_0^2)\mu_0 = C\mu_0 \\
\sum_{x \in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{1-\alpha}{2} (\mathcal{N}(x| - \mu_0, \sigma_0^2) - \mathcal{N}(x|\mu_0, \sigma_0^2))x = C\mu_0 \\
\frac{1}{C} \sum_{x \in X} \frac{1}{\mathrm{P}(x|\theta)} \frac{1-\alpha}{2} (\mathcal{N}(x| - \mu_0, \sigma_0^2) - \mathcal{N}(x|\mu_0, \sigma_0^2))x = \mu_0
\end{aligned} \tag{4.25}$$

Since it is hard to solve Equation 4.25 analytically in closed form due to the existence of μ_0 in both exponential form and denominator, we refer to EM-style iterative algorithm as described in [16] to find local optimal solution for μ_0. We start with a random initialization of the μ_0, then repeatedly update the parameter using their value in the previous iteration. To be exact, μ_0's values on RHS of Equation 4.25 is the value computed from the previous iteration. Later, we shall see that this strategy also applies to learn other parameters.

Derivation of $\mathcal{L}$ w.r.t to α

Denote $\alpha_1 = \alpha$ and $\alpha_0 = 1 - \alpha$. The likelihood function for one data point in Equation 4.6 is transformed to Equation 4.26.

$$\mathrm{P}(x|\theta) = \alpha_1 \mathrm{P}(x|y=1) + \alpha_0 \mathrm{P}(x|y=0) \tag{4.26}$$

Equate the derivatives of Equation 4.22 with respect to α_1, α_0, and λ_α to zero, we have following conditions.

- *Condition 4.1*: Set derivative of Equation 4.22 w.r.t. α_1 to zero:

$$\begin{aligned}\frac{\partial}{\partial\alpha_1}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial\alpha_1}\ln \mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial\alpha_1}\mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\mathrm{P}(x|y=1) + \lambda_\alpha = 0 \end{aligned} \tag{4.27}$$

- *Condition 4.2*: Set derivative of Equation 4.22 w.r.t. α_0 to zero:

$$\begin{aligned}\frac{\partial}{\partial\alpha_0}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial\alpha_0}\ln \mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial\alpha_0}\mathrm{P}(x|\theta) + \lambda_\alpha \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\mathrm{P}(x|y=0) + \lambda_\alpha = 0 \end{aligned} \tag{4.28}$$

- *Condition 4.3*: Set derivative of Equation 4.22 w.r.t. λ_α to zero:

$$\frac{\partial}{\partial\lambda_\alpha}\mathcal{L} = \alpha_1 + \alpha_0 - 1 = 0 \tag{4.29}$$

- *Condition 4.4*: Condition of Lagrange multiplier.

$$\lambda_\alpha \geq 0 \tag{4.30}$$

Multiplying α_1 into both sides of Equation 4.27, and α_0 into both sides of Equation 4.28, and summing them together, with the condition in Equation 4.29, we have: $\lambda_\alpha = -|X|$.

In the **E-step**, we compute $d(x) = \frac{\alpha_1}{\mathrm{P}(x|\theta)}\mathrm{P}(x|y=1)$. In the **M-step**, we

compute α_1 according to Equation 4.31.

$$\alpha_1 = \frac{1}{|X|} \sum_{x \in X} d(x) \tag{4.31}$$

Similarly to the learning of μ_0 above, we also use iterative algorithm to learn α_1. Specifically, the α_1's value in $d(x)$ is taken from previous iteration. At the beginning, the value is randomly initialized.

Derivation of $\mathcal{L}$ w.r.t σ_1 and σ_0

Equate derivatives of Equation 4.22 with respect to σ_1, σ_0, s^2 and λ_σ to zero, we have following conditions.

- *Condition 4.5*: Set derivative of Equation 4.22 with respect to the slack variable s to zero:

$$\frac{\partial}{\partial s}\mathcal{L} = -2\lambda_\sigma s = 0 \tag{4.32}$$

- *Condition 4.6* Set derivative of Equation 4.22 with respect to the multiplier λ_σ to zero:

$$\begin{aligned} \frac{\partial}{\partial \lambda_\sigma}\mathcal{L} &= \sigma_0 - \sigma_1 - s^2 = 0 \\ \sigma_0 &= \sigma_1 + s^2 \end{aligned} \tag{4.33}$$

- *Condition 4.7*: Set derivative of Equation 4.22 with respect to σ_1 to zero:

$$\begin{aligned}\frac{\partial}{\partial\sigma_1}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial\sigma_1}\ln \mathrm{P}(x|\theta) - \lambda_\sigma \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial\sigma_1}\mathrm{P}(x|\theta) - \lambda_\sigma \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\alpha\frac{\partial}{\partial\sigma_1}\left(\frac{1}{\sigma_1\sqrt{2\pi}}\phi_1\right) - \lambda_\sigma \\ &= \sum_{x\in X}\frac{\alpha\mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)}\left(1-\frac{x^2}{\sigma_1^2}\right) - \lambda_\sigma = 0 \end{aligned} \tag{4.34}$$

- *Condition 4.8*: Set derivative of Equation 4.22 with respect to σ_0 to zero:

$$\begin{aligned}\frac{\partial}{\partial\sigma_0}\mathcal{L} &= \sum_{x\in X}\frac{\partial}{\partial\sigma_0} ln\mathrm{P}(x|\theta) + \lambda_\sigma \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}\frac{\partial}{\partial\sigma_0}\mathrm{P}(x|\theta) + \lambda_\sigma \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}(1-\alpha)\frac{\partial}{\partial\sigma_0}\frac{1}{2\sigma_0\sqrt{2\pi}}\left(\phi_0^+ + \phi_0^-\right) + \lambda_\sigma \\ &= \sum_{x\in X}\frac{1}{\mathrm{P}(x|\theta)}(1-\alpha)\left(-\frac{1}{\sigma_0}\mathrm{P}(x|y=0)\right. \\ &\quad \left. + \frac{1}{2\sigma_0^3}\left(\mathcal{N}(x|-\mu,\sigma_0^2)(x+\mu)^2 + \mathcal{N}(x|\mu,\sigma_0^2)(x-\mu)^2\right)\right) + \lambda_\sigma = 0 \end{aligned} \tag{4.35}$$

Denote $D = \sum_{x\in X} d(x)$ as the quantity computed previously to obtain α_1. Set Equation 4.34 to zero to compute σ_1 in the **M-step**, we have Equation 4.36.

$$\begin{aligned}\sum_{x\in X}\frac{\alpha\mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)} &= \sum_{x\in X}\frac{\alpha\mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)}\frac{x^2}{\sigma_1^2} \\ \sigma_1^2 &= \frac{1}{D}\sum_{x\in X} d(x)\cdot x^2 \end{aligned} \tag{4.36}$$

Denote E as the quantity computed in the E-step to compute σ_0.

$$E = \sum_{x \in X} (e_1(x) + e_2(x)) \tag{4.37}$$

$$e_1(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\Theta)} \mathcal{N}(x| - \mu_0, \sigma_0^2) \tag{4.38}$$

$$e_2(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\Theta)} \mathcal{N}(x|\mu_0, \sigma_0^2) \tag{4.39}$$

Set the Equation 4.35 to zero, we have Equation 4.40.

$$\begin{aligned}
\sum_{x \in X} \frac{(1-\alpha)\mathrm{P}(x|y=0)}{\mathrm{P}(x|\theta)} &= \frac{1}{2\sigma_0^2} \sum_{x \in X} \frac{(1-\alpha)}{\mathrm{P}(x|\theta)} \left((\mathcal{N}(x| - \mu_0, \sigma_0^2)(x+\mu_0)^2 \right. \\
&\qquad \left. + \mathcal{N}(x|\mu_0, \sigma_0^2)(x-\mu_0)^2 \right) \\
\sigma_0^2 &= \frac{1}{2E} \sum_{x \in X} \frac{(1-\alpha)}{\mathrm{P}(x|\theta)} \left(\mathcal{N}(x| - \mu_0, \sigma_0^2)(x+\mu_0)^2 \right. \\
&\qquad \left. + \mathcal{N}(x|\mu_0, \sigma_0^2)(x-\mu_0)^2 \right) \\
\sigma_0^2 &= \frac{1}{E} \sum_{x \in X} (e_1(x) \cdot (x+\mu_0)^2 + e_2(x) \cdot (x-\mu_0)^2))
\end{aligned} \tag{4.40}$$

Substitute Equation 4.33 into the right hand side of Equation 4.40 to compute s^2 using the old value of σ_1, then update the new value of σ_0 with newly computed s using Equation 4.41 in the **M-step**.

$$\sigma_0 = s + \sigma_1 \tag{4.41}$$

We summarize the computations in the E-step and M-step in the following. Both steps are computed iteratively till convergence.

In the **E-step**, we compute the following quantities (to be used in the next M-step):

- $c(x) = \frac{1-\alpha}{2\mathrm{P}(x|\theta)}(\mathcal{N}(x| - \mu_0, \sigma_0^2) + \mathcal{N}(x|\mu_0, \sigma_0^2))$

- $d(x) = \frac{\alpha \mathrm{P}(x|y=1)}{\mathrm{P}(x|\theta)}$
- $e_1(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\theta)}\mathcal{N}(x| - \mu_0, \sigma_0^2)$
- $e_2(x) = \frac{(1-\alpha)}{2\mathrm{P}(x|\theta)}\mathcal{N}(x|\mu_0, \sigma_0^2)$

In the **M-step** we compute μ_0, σ_1, σ_0 and s.

- $\mu_0 = \frac{1}{C}\sum_{x \in X}(e_1(x) - e_2(x))x$, where $C = \sum_{x \in X} c(x)$
- $\alpha = \frac{1}{|X|}\sum_{x \in X} d(x)$
- $\sigma_1^2 = \frac{1}{D}\sum_{x \in X} d(x) \cdot x^2$, where $D = \sum_{x \in X} d(x)$
- $\sigma_0 = (\frac{1}{E}\sum_{x \in X}(e_1(x) \cdot (x + \mu_0)^2 + e_2(x) \cdot (x - \mu_0)^2))^{-\frac{1}{2}} + \sigma_1$, where $E = \sum_{x \in X}(e_1(x) + e_2(x))$

Once the parameters are learned, we can make inferences for the posterior probability of agreement $\mathrm{P}(y = 1|x)$, based on Equation 4.5, and substituting the learned parameters θ.

4.3 Experiments

Here we first examine how *Global* and *Local* fit the distribution of rating difference. We also show an application of the combined method *CAM-DPMF* to generate contextual agreement probabilities in Section 4.3.3.

4.3.1 Experimental Setup

Datasets. We conduct experiments on the preprocessed rating data sets described in Section 3.3. For each data set, we create two sets of training/testing data using the same process in Section 3.3, namely rating difference data and rating data. Since the *Local* model (Section 4.2) works on the user pair level, we apply five-fold cross-validation on each pair (u, v)'s observed data with ratio of 80/20 for training/testing set to ensure the pairs have all rating difference

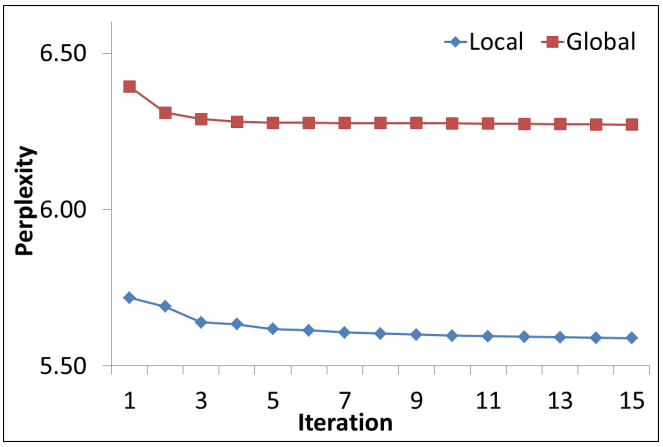


Figure 4.3: Perplexity of *CAM* on Ciao Testing Set

instances in both training/testing set. For *Global*, we simply combine the training/testing set of all pairs in the same fold to create a large training/testing sample, X_{train} and X_{test}. Each experiment is run five times on each of the five folds. The final result is reported as the average over the 25 runs for each setting.

We also apply the same process in Section 3.3 to create R_{train} and R_{test} from X_{train} and X_{test}.

4.3.2 Experimental Results

First, we study the parameter learning for *CAM*. As mentioned in Section 4.2.1, the parameters for *CAM* can either be *Global* (same θ for all user pairs), or *Local* (specific θ_{uv} to each user pair). One measure of effectiveness for a probabilistic model is *perplexity*, or the ability of model parameters learned from training data (X_{train}) to fit the testing data (X_{test}). Equation 4.42 shows that it is measured similarly as in [18], where $p(x_{uvi})$ is the likelihood of observing x_{uvi} as shown in Equation 4.6. Lower perplexity is better, as it indicates a higher likelihood of observing the unseen data.

$$perplexity(X_{test}) = exp\left\{-\frac{\sum_{x_{uvi} \in X_{test}} \log p(x_{uvi})}{|X_{test}|}\right\} \tag{4.42}$$

In Figure 4.3, we plot the log-likelihood achieved by *Global* vs. *Local* over

iterations on the Ciao dataset. It shows that convergence is attained relatively swiftly in just a few iterations. For this reason, the EM algorithms' stopping condition was set to 15 iterations. Similar trends are observed across all datasets. The perplexity after 15 iterations are shown in Table 4.1. For all seven datasets, *Local* has lower (better) perplexity than *Global*. This result is expected as each user pair has a distinct behavior, and the global parameter will not fit all pairs equally well. Assigning each pair a unique set of parameters can capture their co-rating behaviour better.

Table 4.1: Perplexity of CAM on Testing Set (statistically significant best-performing entries are asteriated)

Dataset	Local	Global
Ciao	5.59*	6.27
Epinions	5.12*	5.38
Flixster06	4.99*	5.23
Flixster07	4.86*	5.15
Flixster08	4.49*	4.80
Flixster09	4.45*	4.75
Movielens100K	5.31*	5.56

Although *Global* has worse perplexity in general, it may still have advantages over *Local* on some pairs with very few observations to learn from. In order to verify our hypothesis, we partition user pairs into bins according to the total number of observed rating differences. Each bin is represented as a range (a, b]. In each bin, we record the fraction of pairs for which *Local* has better fit than *Global*. The same process is applied across the five folds of each dataset. Table 4.2 reports the average percentages. The common observation across all datasets is that the fraction of pairs for which *Local* has better fit is consistently decreasing with the number of observation. For Flixster09, the last bin has very few instances, which may explain why it is an exception to the general trend. *Global* indeed is more likely to perform better than *Local* on pairs with fewer observations than on pairs with many observations. However, since *Local* has better performance in general (i.e., all the fractions are greater than 50%), we will use *Local* for *CAM* in subsequent experiments.

Table 4.2: Ratios of number of pairs in a bin that *Local* returns higher perplexity than *Global*.

	No. of observations			
Dataset	≤ 50	(50, 100]	(100, 150]	> 150
Ciao	78%	82%	88%	93%
Epinions	70%	73%	77%	84%
Flixster06	71%	72%	79%	86%
Flixster07	68%	75%	80%	83%
Flixster08	70%	77%	80%	84%
Flixster09	70%	75%	85%	75%
Movielens100K	71%	76%	78%	88%

The EM learning algorithms are relatively efficient. For *Global*, for each fold, convergence is achieved within 1 second for all datasets on an Intel(R) Xeon(R) Processor E5-2667 2.90GHz machine. For *Local*, for each fold, the parameters for all consumer pairs can be learned in 1 to 4 minutes.

4.3.3 Application: Similarity-based Neighborhood Collaborative Filtering

Here, we use the model parameters of *CAM*, combined with the rating difference predictions by *DPMF* to generate contextual agreement probabilities $w_{uvi} = \mathrm{P}(y_{uvi}|\hat{x}_{uvi})$. These probabilities are used in a user-based nearest-neighbor collaborative filtering [44]. In the rating prediction task, the technique exploits the similarities between consumers to predict unseen ratings. For every rating $r_{ui} \in R_{test}$, we predict $\hat{r}_{ui}$ as a weighted average of neighbors' ratings in R_{train}. Neighbor v can be any user, weighted by w_{uvi}.

$$\hat{r}_{ui} = \frac{\sum_{v \neq u, r_{vi} \neq \phi} w_{uvi} \times r_{vi}}{\sum_{v \neq u, r_{vi} \neq \phi} w_{uvi}} \tag{4.43}$$

The accuracy of rating prediction is measured by $RMSE_{rating}$ defined in Equation 4.44.

$$RMSE_{rating} = \sum_{r_{ui} \in R_{test}} \sqrt{\frac{(\hat{r}_{ui} - r_{ui})^2}{|R_{test}|}} \tag{4.44}$$

Table 4.3: Versus Shared Preference ($RMSE_{rating}$, statistically significant best-performing entries are asteriated)

		Shared Preference		
Dataset	CAM-DPMF	Uniform	Cosine	Pearson
Ciao	0.76*	1.15	1.14	1.14
Epinions	0.81*	1.06	1.06	1.06
Flixster06	0.95*	0.98	0.98	0.98
Flixster07	0.90*	0.98	0.95	0.95
Flixster08	0.90*	0.96	0.96	0.95
Flixster09	0.86*	0.93	0.92	0.91
MovieLens100K	0.83*	1.02	1.02	1.02

Note that what is being evaluated here is the weights w_{uvi}'s, and not the rating prediction method. Therefore, it is not our goal to compare to all rating prediction methods. Instead, the reasonable evaluation is to fix the prediction method to neighborhood-based collaborative filtering, and vary the weights based on various baselines. What we consider baselines here are other approaches that also depend on similarity or agreement between a pair of consumers. We include two most commonly used similarity measures, namely Cosine similarity and Pearson's correlation.

Contextual vs. Shared. We compare the efficacy of contextual agreement (labeled *CAM-DPMF*) as compared to baselines relying on shared preference that applies to all items of the same user pair as measured by Pearson and Cosine functions (see Section 2.1). We also include another baseline, called *Uniform*, which is the simple average of the ratings by neighbors, assuming all neighbors are considered to have the same similarity value. The prediction accuracies in terms of $RMSE_{rating}$ are listed in Table 4.3. For all of the datasets, *CAM-DPMF* has the lowest errors. As all the comparative methods work with exactly the same set of ratings, the only difference is how each method weighs the contribution of each rating. This result shows that paying attention to context, as *CAM-DPMF* does, helps to gain a lower prediction error.

***CAM-DPMF* vs. Components**. Since *CAM-DPMF* is a combination of *CAM* and DPMF, we now evaluate the efficacy of the individual components

Table 4.4: Versus Components (RMSE_{rating}, statistically significant best-performing entries are asteriated)

		Components	
Dataset	CAM-DPMF	CAM-α	Linear-DPMF
Ciao	0.76*	1.12	0.82
Epinions	0.81*	1.04	0.86
Flixster06	0.95*	0.97	0.95
Flixster07	0.90*	0.94	0.90*
Flixster08	0.90*	0.93	0.91
Flixster09	0.86*	0.90	0.87*
MovieLens100K	0.83*	1.00	0.86

alone in the rating prediction task. *CAM-α* uses the α_{uv} of each pair as a shared similarity value. For the *DPMF* on its own, we linearly transform the predicted x_{uvi} into a similarity value as follows, where RD_{max} is the maximum possible value of rating differences in the training set. We call this approach as *Linear-DPMF*.

$$w_{uvi} = 1 - \frac{|\hat{x}_{uvi}|}{RD_{max}} \tag{4.45}$$

Table 4.4 shows that the combined approach *CAM-DPMF* achieves the lowest error rates, which supports the necessity of integrating the two components to capture different aspects of the data. One interesting observation is that both *CAM-α* and *Linear-DPMF* also consistently perform better than conventional similarity measurements such as Cosine or Pearson (statistically significant at 0.01). It shows the synergy when combining the ability of predicting unseen rating differences of *DPMF* with the ability of modelling user-pair agreement of *CAM*. Meanwhile, the relatively good performance of *DPMF*, though still worse than the combined *CAM-DPMF*, shows the important role of the former in contributing to the latter's performance.

4.4 Discussion

4.4.1 Analysis of Prevalence of Agreement

In this section, we analyze the prior probability of agreement α_{uv}'s (for different user pairs) for various datasets. This parameter is the prior probability of agreement $\mathrm{P}(y_{uvi} = 1)$ for a pair of consumers u and v. In particular, we are interested in how these probabilities vary across user pairs, depending on varying attributes, such as friendship, demographics, and time.

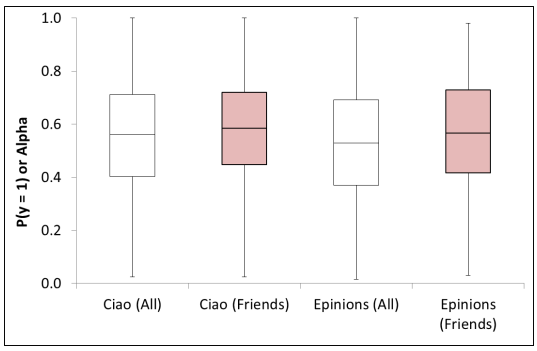


Figure 4.4: Distributions of $\mathrm{P}(y_{uvi} = 1)$ or α_{uv} with 'friendship' factor.

Friendship. First, we test the frequently made hypothesis that friendship or trust relationship can help in learning the preferences of consumers [71, 70]. This analysis could only be performed on *Ciao* and *Epinions* datasets. $MovieLens100K$ does not have social network information. *Flixster* after filtering does not contain sufficient number of social links for statistical tests.

In Figure 4.4, we draw the distributions of α_{uv}, for two populations. The first, drawn in white, concerns all consumer pairs. The second, drawn in red, narrows down the population to only those user pairs sharing friendship or trustor-trustee relationship. One observation is that friendship does contain some information. The comparison of every pair of white (all pairs) vs. red (friends-only) box plots, show that friends have greater agreement (statistically significant) in general. Another interesting observation is that even some friends disagree a lot, as shown by the lower whiskers of the box plots. Hence, just because a pair of consumers are friends, it does not mean they always

agree. Therefore, it is helpful to know the context of agreement.

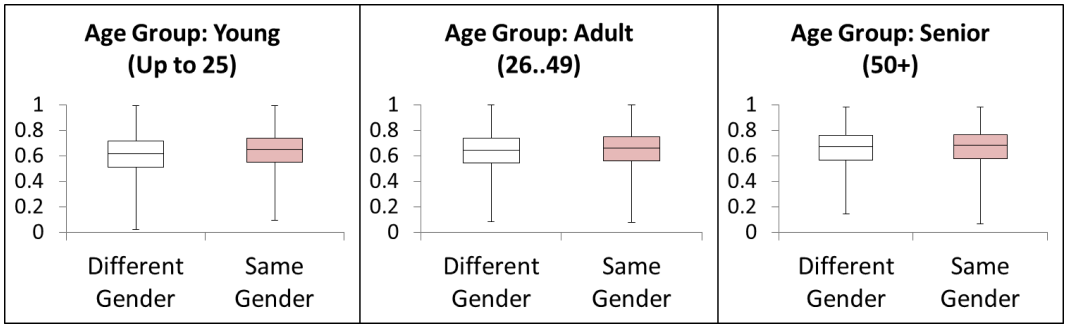


Figure 4.5: Distribution of α_{uv} across different age groups and genders on MovieLens100K.

Demographics. Since *MovieLens*100 contains demographic information, we study whether age and gender has an effect on the probability of contextual agreement. We split the consumers into three different age groups, i.e., Youth (up to 25), Adult (from 26 to 49) and Senior (above 50). Within each age group, we compare whether user pairs of the same gender (both males or both females) would have greater similarity than different genders. Figure 4.5 shows that within the age groups 'Young' and 'Adult', same-gender user pairs have higher agreement preferences (statistically significant) than pairs of different gender. The effects of gender on the 'Senior' age group is much weaker. Overall, demographics do not seem to have as much an effect as friendship does.

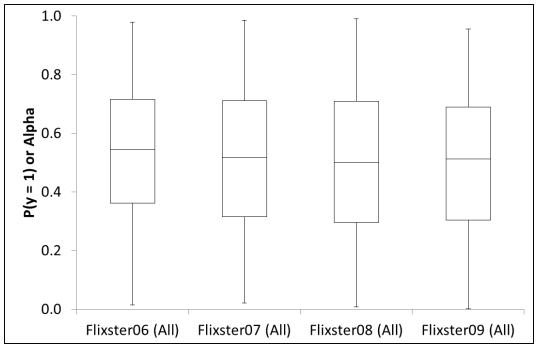


Figure 4.6: Distributions of $\mathrm{P}(y_{uvi} = 1)$ or α_{uv} with 'time' factor.

Time. Since Flixster can be split into four datasets for different years 2006, 2007, 2008, and 2009, we are also interested in the variation across time. In Figure 4.6, we plot the distribution of α_{uv} for each Flixster subset containing data belong to a specific year. The plots show that user agreement generally

remains stable across the years. We perform paired-sample t-tests on user pairs for two consecutive years each time. The variance across years is statistically insignificant during the three years 2007, 2008 and 2009, except for Flixster06, which has somewhat higher agreement than Flixster07. Since Flixster is about movies, we hypothesize that the heterogeneity of movie themes across years may account partially for the variance in agreement.

4.4.2 Case Study

To illustrate the workings of *CAM*, we now show a case study drawn from the MovieLens100K dataset, involving the same pair of consumers as in Section 1.1.1. Table 4.5 shows the ratings of user u (u_{38}) and v (u_{197}) on twenty movies. Based on these ratings, the *CAM* parameters for this pair are as follows: $\alpha = 0.28$, $\mu_0 = 2.09$, $\sigma_0 = 1.36$, $\sigma_1 = 1.48$. The relatively low α suggests that this pair do not always agree. That $\mu_0 = 2.09$ suggests that when they disagree their rating difference is around 2. This is evident from the fourth column labeled $|x_{uvi}|$, which tracks their rating differences. *CAM* uses these parameters to estimate the contextual probability of agreement shown in the fifth column. As expected, the probability of contextual agreement is high (close to 1) for the movies in the shaded middle of the table (where rating differences are low), and is low (close to 0) for the other movies. In contrast to the item-specific agreement produced by *CAM*, the baselines Pearson and Cosine each assign a single similarity value that applies to all items, inadequately describing the nature of agreement between consumers. However, we note that this case study shows a single case, and is meant to be illustrative, and not comparative. The comparative analyses across many user pairs in aggregate were conducted in the earlier Sections 3.3.3 and 4.3.3.

Table 4.5: MovieLens Case Study

Movie	r_{ui}	r_{vi}	$\lvert x_{uvi}\rvert$	CAM	Pearson	Cosine
Conan the Barbarian	5	1	4	0.05		
Volcano	5	2	3	0.16		
First Knight	5	2	3	0.16		
Scream	5	3	2	0.41		
G. I. Jane	5	3	2	0.41		
George of the Jungle	3	1	2	0.41		
Titanic	5	4	1	0.80		
Liar Liar	5	4	1	0.80		
Top Gun	5	4	1	0.80		
Braveheart	5	5	0	1.00	-0.26	0.81
Jurassic Park	5	5	0	1.00		
Conspiracy Theory	4	4	0	1.00		
Die Hard (1995)	2	3	1	0.80		
Full Metal Jacket	2	3	1	0.80		
The Fugitive	3	5	2	0.41		
Batman (1989)	1	3	2	0.41		
The Godfather	2	5	3	0.16		
Die Hard 2 (1990)	1	4	3	0.16		
Ben Hur	1	5	4	0.05		
The Terminator	1	5	4	0.05		

4.4.3 Summary

In this chapter we discuss the problem of estimating the degree of agreement in preferences between two consumers on a particular product. Our observation is that rating differences can signal how two consumers agree in their preferences. We propose *CAM*, a Gaussian mixture model with special constraints on the parameters, to measure the agreement in preferences from rating differences. We show that the combination *CAM-DPMF* brings better predictive performance than the common similarity measures in an application of User-based Collaborative Filterings. Moreover, analysis on *CAM*'s parameters reveals characterisitics of the preference agreement in specific consumer segments.

Part II

Diversity of Preferences between Products across Various Consumers

Chapter 5

Estimating Willingness-to-pay from Online Reviews

5.1 Overview

Recall from Section 1.1.2 that the main focus in this Part II is on computational challenges in bundling products based on consumers' willingness-to-pay. Estimating willingness to pay is complex, and encompasses a whole research area [21]. To demonstrate the efficiency of the proposed solutions in Chapters 6 and 7, we advocate a systematic method to derive willingness-to-pay at a large scale (many consumers and many products).

Consumers often express their preferences through online reviews. Specifically, they rate products according to their preferences. Higher ratings often imply more satisfaction. Besides, consumers would discuss or share how they enjoy product features by writing reviews. Due to its availability at large volume, there have been studies focusing on automated tasks to elicit consumer preferences from online review data [8, 65]. For example, [65] proposes text mining techniques to mine consumer feedback on product features from their reviews. [8] studies the correlation between consumer preferences extracted from their reviews and product sales.

For its availability and reflectiveness of consumer preferences, we propose two approaches to elicit consumers' willingness-to-pay from their online reviews. The first approach is based on an intuition that consumers are willing to pay more for their preferred products. Since their preferences are reflected through ratings, we propose to estimate willingness-to-pay from rating data in Section 5.1.1. For the second approach, we rely on the concept of product utility as mentioned in Section 2.1. [46] proposes a transformation from product utility to willingness-to-pay. Estimating product utility requires consumers' ratings on products, as well as their feedback on the feature basis [21]. Since both information can be extracted from online reviews, we propose a framework in Section 5.1.2 to elicit willingness-to-pay from online reviews.

5.1.1 Rating-based Willingness-To-Pay

We now formulate the problem of estimating willingness-to-pay from ratings.

Notations. Given the consumer set $\mathcal{U}$, product set $\mathcal{I}$. By u and i we refer to a consumer and a product in the given sets. By p_i we denote the price of i. Denote r_{ui} as the rating that u gave to i. $r_{ui} = 0$ if the rating is missing in our data. Otherwise, $r_{ui} > 0$.

Problem formulation. Given a consumer u, a product i, the rating r_{ui}, and the product price p_i, the task is to estimate how much u is willing to pay for i.

Linear transformation. We propose a linear transformation to convert willingness-to-pay from rating. For a rating $r_{u,i}$ of consumer u on product i, the consumer is willing to pay at most $w_{u,i}$ for the product.

$$w_{u,i} = \frac{r_{u,i}}{r_{max}} \times \lambda \times p_i, \tag{5.1}$$

where r_{max} is the maximum rating possible, p_i is the price of product i, and λ is the conversion rate. In this work we set $\lambda = 1.25$, $r_{max} = 5$.

Although this rating-based approach is easy to implement and requires only rating data, estimating willingness-to-pay using Equation 5.1 does not capture the trade-off between preferences on product features and the price. Hence it also does not provide explanation on the estimated values. To overcome this issue, we rely on both ratings and review text as shown in the following section.

5.1.2 Review-based Willingness-To-Pay

For the second approach, we present the Preference Elicitation framework to extract willingness-to-pay from both ratings and review text. The framework consists of several components. First, we extract product price, product features as well as consumers' feature-wise feedback from the given data. Based on the extracted information, we conduct Conjoint Analysis [21] to estimate the overall utility obtained by consumer u from consuming product i as mentioned in Section 2.1. The estimated utility is fetched into the **Willingness-to-Pay Extraction** component to produce consumers' willingness-to-pay.

By $U(u, i)$ we denote the utility obtained by u from consuming i. To estimate $U(u, i)$ from product rankings and consumers' valuations on the feature basis, Conjoint Analysis employs Equation 2.1 with a slight modification as described in Section 5.2. Although rankings among products can be induced directly from ratings, it is not that straightforward to identify feature-level valuations from online reviews. In our framework, the number of features K and label for each feature coefficient q_{ik} are first identified in the **Feature Identification** component and then served as inputs to the **Feature Evaluation** for estimating the corresponding consumer valuations in their reviews. Outputs from these two components are sufficient to perform Conjoint Analysis to estimate the desired utility.

With the indispensable role of Conjoint Analysis in the framework, we coin the whole process above as *Conjoint Analysis on review data*. Details of each component are reported in Section 5.2.

Notations. Given the consumer set $\mathcal{U}$, product set $\mathcal{I}$. By u and i we refer to a consumer and a product in the given sets. By p_i we denote the price of i. Every u is associated with a set of reviews $\mathcal{R}_u$. Each review $\mathcal{R}_{ui}$ comprises of a sequence of sentences and a rating score in scale of 5. Denote $\boldsymbol{r}_u = \{r_{ui} | \mathcal{R}_{ui} \in \mathcal{R}_u\}$ the list of scores associated with the review set $\mathcal{R}_u$. Denote the total number of features extract from all $\mathcal{R}_u$ as K.

Let us consider a choice set $\mathcal{P}_u$, $\mathcal{P}_u = \{(i,j)_1, (i,j)_2, \ldots\}$, where each l-th choice associates two products i and j in $\mathcal{I}$. Without loss of generality, we assume u prefers i to j. Moreover, we associate each consumer u with a vector $\boldsymbol{S}_u \in \mathbb{R}^K$, and every product i with a vector $\boldsymbol{Q}_i \in \mathbb{R}^K$. For every product i in $\mathcal{P}_u$, by $\boldsymbol{V}_{ui} = \text{diag}(\{v_{ui1}, \ldots, v_{uiK}\})$ we denote the diagonal matrix projecting u's preferences into i's feature space.

Problem Formulation. Given a review set $\mathcal{R}$ of a consumer, a set of product $\mathcal{I}$ and their price, the task is to estimate how much the consumer is willing to pay for a product based on her review $\mathcal{R}_i$.

5.2 Methodology

In this section we describe the framework to estimate consumers' willingness-to-pay from their online reviews. Recall from Section 5.1 that at the heart of the framework is applying Conjoint Analysis on review data. The whole process consists of four steps as illustrated in Figure 5.1.

- **Step 1: Feature identification.** Given a review set of a product category, the first step is to extract frequently mentioned product features (Section 5.2.1). Intuitively, these features can be identified through common words across reviews. This intuition motivates a two-phase approach. We first employ a clustering technique to extract groups of highly co-occurred popular words. Product features are later identified from examining top frequent words in each group.

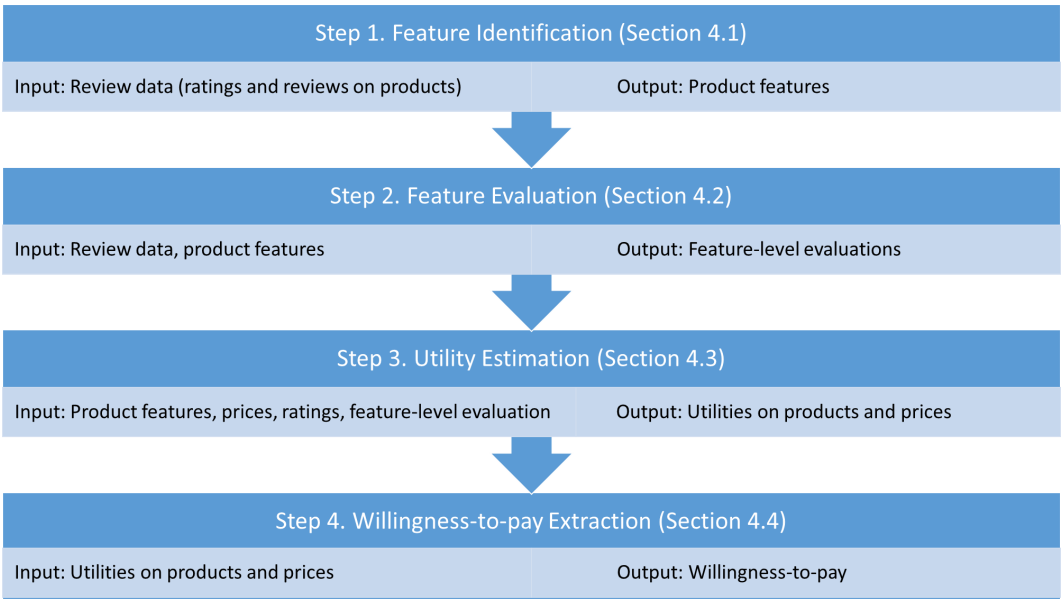


Figure 5.1: The transformation process to extract willingness-to-pay from review data.

- **Step 2: Feature evaluation.** Given the feature set from Step 1, we measure how a consumer values each feature based on his or her reviews. We employ three different evaluation schemes, namely Binary, Frequency and Sentiment. Details on each scheme can be found in Section 5.2.2. After this step, for each consumer's review we have a list of features mentioned in the review with the corresponding evaluation.

- **Step 3: Utility estimation.** For a given consumer with his/her reviews, we now have sufficient "ingredients" to perform Conjoint Analysis in order to estimate the consumer's utilities on reviewed products. The input consists of a list of ranked products according to their rating, a set of features and their corresponding evaluation and product prices. The technical underpinning of the analysis is the optimization problem in Equation 5.12. By solving the optimization problem, we can derive the desired utilities of the consumer on his or her reviewed products.

- **Step 4: Willingness-to-pay extraction.** The final step described in Section 5.2.4 is to transform the estimated utilities in Step 3 into

willingness-to-pay.

In the following, we first discuss the preference model underlies the Conjoint Analysis. The model motivates the four-step framework above. More details on each step are discussed in the subsequent sections.

The theoretical preference model. Suppose that we observe a choice set $\mathcal{P}_u$ for every consumer u. Each product $i \in \mathcal{P}_u$ is associated with a set of v_{uik}'s, representing how u would value the k-th aspect of i. Conjoint Analysis aims to estimate the utility $U(u, i)$ for every $i \in \mathcal{P}_u$ satisfying for a pair $(p_l^1, p_l^2) \in \mathcal{P}_u$, $U(u, p_l^1) > U(u, p_l^2)$.

There are several assumptions. For one thing, similar to Section 2.1, the analysis also assumes $U(u, i)$ is a linear combination of part-worth utilities. Beyond that, each v_{uik} is assumed to contribute to the corresponding part-worth utility. Therefore, the overall utility is slightly different than Equation 2.1.

$$U(u, i) = \sum_k s_{uk} \times v_{uik} \times q_{ik} \tag{5.2}$$

Since all product i share the same set of features, each q_{ik} is set to 1. Recall that $\boldsymbol{V}_{ui} \in \mathbb{R}^{K \times K}$ is a diagonal matrix whose diagonal entries are v_{uik}. We can re-formulate the Equation 5.2 as follows

$$U(u, i) = \boldsymbol{S}_u^T \boldsymbol{V}_{ui} \boldsymbol{Q}_i \tag{5.3}$$

It is straightforward that Equation 2.1 is a special instance of Equation 5.3 when $\boldsymbol{V}_{ui}$ is an identity matrix. Since $\boldsymbol{V}_{ui}$ is not given as inputs, all the features are assumed to receive equal valuations from the consumer. When $\boldsymbol{V}_{ui}$ is observed, $\boldsymbol{S}_u$ is considered to regulate the weighted contribution of u's valuations on each feature to the overall utility.

Toward the goal of estimating willingness-to-pay, the Conjoint Analysis requires information of product price as part of the model. Besides K features, each product i is associated with a price $price_i$. The product price is

also assumed to bring some utility to consumers, but at a loss rather than a gain [63]. Hence, a common setting in choice-based Conjoint Analysis studies [46, 29] is to integrate product price into the utility function as follow: $U(u, i, price_i) = \sum_{k=1}^{K} s_{uk} v_{uik} + s_{uprice} price_i$. Since $\boldsymbol{V}_{ui}$ and $price_i$ are observed, seeking a desired $U(.)$ is equivalent to solving for a set of parameters $\boldsymbol{S} = \{s_1, \ldots, s_K, s_{uprice}\}$ satisfying all the pairwise rankings in $\mathcal{P}$

$$
\begin{aligned}
U(u, i, price_i) &\geq U(u, j, price_j), \forall (i, j) \in \mathcal{P} \\
\sum_{k=1}^{K} s_{uk} v_{uik} + s_{uprice} price_i &\geq \sum_{k=1}^{K} s_{uk} v_{ujk} + s_{uprice} price_j, \forall (i, j) \in \mathcal{P}
\end{aligned} \tag{5.4}
$$

To construct the inequality system in Equation 5.4, we require two pieces of information for every consumer: 1) her choice data $\mathcal{P}$ and 2) her feature valuations $\boldsymbol{V}_{ui}, \forall i \in \mathcal{P}$. There are two common practices to create the preference data. The first approach is to use Monte Carlo simulations [29], for instance sampling $\boldsymbol{V}_{ui}$ from a Gaussian distribution with controlled mean and variance. Another approach is to extract these information from review data. In [48], the authors show a method to transform rating data to pairwise choice data. Meanwhile, [8] leverages text processing techniques such as text clustering, syntactic dependency parser, etc., to obtain $\boldsymbol{V}_{ui}$ from reviews. Inspired from these two works, we briefly overview the following approaches to obtain both $\mathcal{P}$ and $\boldsymbol{V}_{ui}$ for every consumer from her online reviews.

Given a review set $\mathcal{R}$, we can extract $\boldsymbol{r}$ accordingly. Each review $\mathcal{R}_i$ about the product i includes a sequence of sentences and a rating score. Each response type reflects consumer evaluation on different levels. Rating score conveys consumer satisfaction in general. The higher the score is, the more satisfied the consumer feels about consuming the product. A usual rating scale is either binary (like/dislike) or ordinal (e.g. 1 to 5 or 1 to 10). As the scores indicate

relative preferences among products, we can construct P accordingly.

$$\mathcal{P} = \{(i, j) | \forall i, j : r_{ui} > r_{uj}\} \tag{5.5}$$

Note that two products having the same scores are not included in $\mathcal{P}$.

On the other hand, consumers would elaborate their ratings in words. Each sentence in a review may address one or more features, or none of them. There are certain words or phrases to describe a feature, such as weight, color, etc, which are widely-used in different reviews. By capturing highly frequent terms appearing in the review set, we can identify most concerned product features from consumers' viewpoints. This intuition motivates our selection of a clustering technique on text data in Section 5.2.1. Assume that there are K features identified from the technique. To create $\mathbf{v}_{ui}$, we propose three different evaluation schemes in Section 5.2.2.

Given the inequality system in Equation 5.4, to learn the utility function, we formulate and solve an optimization problem in Equation 5.12 from the preference model in Section 5.2.3. The output utilities are then transformed to consumers' willingness-to-pay in Section 5.2.4.

5.2.1 Feature Identification

Extracting product features from reviews is a well-studied problem in Text Mining area [41]. As shown in [8], the existing text mining techniques are capable of extracting high quality product features as human annotators. Hence, we also resort to text mining techniques for automatically extracting features from review data. Our approach is based on the following observations of how product features are frequently mentioned in reviews.

A product feature can be described by a set of words. If a sentence in a review mentions a product feature, it would contain the feature's representative words. It is possible to have a sentence mentioning more than one features, or

many sentences mentioning a feature.

From the above observations, we can identify a product feature from a set of words which likely occur together in a sentence. However, any set of words do not necessarily address a feature. We postulate that the word sets appearing in many sentences are more likely to address a product feature. It motivates a two-phase approach. At Phase 1, we seek such popular word sets from review sentences. At Phase 2, product features are then identified from the popular word sets. Moreover, we can also identify the features mentioned in each sentence.

To find popular word sets from sentences, we refer to a slew of text clustering algorithms in the literature [4]. Since only the sentences are given as priori, other basic information about those word sets are unknown, such as the occurrence of a popular word set, and the number of popular word sets. We resort to hierarchical Latent Dirichlet Allocation (hLDA), a nonparametric clustering model which can derive these information from the given data [17]. We briefly summarize *hLDA* below.

Hierarchical Latent Dirichlet Allocation (hLDA)

At a glance, hLDA is a non-parametric probabilistic model which is used to cluster text data. There are several core concepts in hLDA.

- From the given corpus of sentences $\mathcal{D}$, hLDA first defines a vocabulary dictionary V, or a set of unique words in D.

- hLDA denotes a topic z as a reference to a probabilistic vector ϕ_z over V. As the vector itself defines a multinomial distribution over V, z can be loosely seen as a word set. Words with high values in ϕ_z can be interpreted that they are often used to described z. Hence, a common practice in literature is to select words with highest ϕ_z values to represent and interpret z [17].

- Since each word in a sentence d may be used in multiple topics, there

can be more than one topics described in d. To represent the likelihood d mentions a specific topic, hLDA uses another probabilistic vector θ_d defining a multinomial distributions over the topic set. Topics with high θ_d values are the ones whose representative words appear more frequently in d.

Conceptually, to derive the number of topics, i.e., word sets, directly from data, hLDA starts with a small number of topics, and spawn new ones when it is necessary to capture a new sentence d. To prevent the number of topics from growing uncontrollably[1], hLDA assumes a hierarchical structure to capture the relationship between topics. The tree-like structure can have many layers, reflecting the degree of generalization/specification of topics. Each layer has many nodes. Each node contains a topic. The layer of the root node is indexed at 0. The closer the layers are to the root (i.e., the smaller the layers are), the more general the topics locate at these layers. For example, the topic at the root node containing stop words. The topics residing at child nodes are more specific to the one locates in the corresponding parent node. For example, one topic at a node is about "iPhone". Each child node of that topic would indicate different generations of each brand such as "iPhone SE", "iPhone 6", etc. To generate the hierarchical structure from a given corpus, hLDA employs nested Chinese Restaurant Process (nCRP) [99].

In order to estimate all the topic-word and sentence-topic probabilistic vectors, hLDA admits a generative process based on the hierarchical structure generated by nCRP to produce a sentence. Each sentence-topic distribution θ_d is defined over a path in the hierarchical structure. To generate a path, from the root note hLDA would decide whether the next node is among the root node's existing child nodes, or a new node using the nCRP. If the next node is

[1]If we do not assume any relationships between topics, the number of topics can grow up to infinity as each document is best described by its own unique set of topics. A large number of topics leads to a sparse distribution of words, which in turn makes those topics more difficult to interpret. For example, each topic may contain only one word, i.e., one word has highest probability mass while others' are negligible.

a new one, then a new node is spawn and linked to the root node. Repeat the similar process on the chosen node until a certain level is reached. Although the structure's depth is limited to a user-specified number, theoretically the structure can grow as many paths as possible to fit the given data. Together with the root node, all the visited nodes form a path in the structure. Given the path, words in d are then sampled using θ_d.

Generative process. We now summarize generative process for each sentence d on the given corpus D. More details can be found in [17].

For every sentence $d \in \mathcal{D}$:

(1). Draw a path $\boldsymbol{c}_d$ from the root node

$$\boldsymbol{c}_d \sim \text{nCRP}(\gamma), \tag{5.6}$$

where γ is the smoothing parameter of the nCRP controlling the probability of spawning new topic.

(2). Draw a distribution over words in V for each topic z in $\boldsymbol{c}_d$

$$\phi_z \sim \text{Dirichlet}(\eta) \tag{5.7}$$

(3). Draw a distribution θ_d over topics in $\boldsymbol{c}_d$

$$\theta_d \sim \text{GEM}(m, \pi), \tag{5.8}$$

where $\text{GEM}(m, \pi)$ is the GEM distribution [80]. Basically, this two-parameter stochastic process returns a probabilistic vector (e.g., sum of elements equals to 1) of any length. m and π control the distribution of mass on the probabilistic vector. High/low value of m would lead to the concentration of mass towards the first/last few entries of the vector. π in turn controls the variance of the mass concentration.

(4). Draw an outcome for each of word token $w_{n,d}$ in d:

(4.1) Draw an outcome for the topic of the n-th word in d:

$$z_n \sim \text{Multinomial}(\theta_d), z_n \in \boldsymbol{c}_d \tag{5.9}$$

(4.2) Draw an outcome for the choice of word in $w_{n,d}$:

$$w_{n,d} \sim \text{Multinomial}(\phi_{z_n}), w_{n,d} \in V \tag{5.10}$$

In [17], the authors propose Gibbs sampling algorithm to infer all the parameters in the model, $\Theta = \{\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{\phi}\}$. Across the Gibbs iterations, the topic-word and sentence-topic assignments may change. Hence, after an iteration, it is possible to have topics which are not assigned to any sentences. We ignore such topics in that iteration.

5.2.2 Feature Evaluation

After Step 1, we can identify the most concerned K feature coefficients with their labels in a category though consumer reviews. We can also identify the subset of features mentioned in a particular review $\mathcal{R}_i$, which is the basis to compute $\boldsymbol{V}_{ui}$. There are three approaches to measure each v_{uik} from reviews, namely **Binary**, **Frequency** and **Sentiment**.

- **Binary** measurement is based on observations that consumers only mention features important to them in their reviews. For each $\mathcal{R}_i$, we set v_{uik} to one if the feature k is mentioned in $\mathcal{R}_i$. Otherwise $v_{uik} = 0$.

- **Frequency** measurement counts the number of times a feature mentioned in a review. The intuition is that the more frequently a feature is mentioned in the product, the more important it is to the consumer. Given $\mathcal{R}_i$, we set v_{uik} to the frequency of the feature k mentioned in $\mathcal{R}_i$. Otherwise $v_{uik} = 0$.

- **Sentiment** measurement is a common approach in text mining area to determine consumer's opinions about the features mentioned in the reviews. The general idea is to locate "sentimental" words, mostly adjectives, associated with the features. We first set all v_{uik} to zero. Then we apply the method in ([93]) to measure the sentiment level for every review sentence. Each review sentence is associated with an ordinal number, indicating the level of positiveness in sentiment. Specifically, given a sentence as input, there are five different scales of sentiment score, ranging from very negative (1), negative (2), neutral (3), positive (4) to very positive (5). If a review sentence discusses more than one features, the sentence's sentiment score is split evenly among the features. Each v_{uik} is then computed as the total scores the product 's k-th feature receives from all the review sentences address it. Table 5.6 shows several examples of the sentimental scores evaluated on review sentences.

5.2.3 Utility Estimation

Given K and feature labels extracted from Step 1 and all $\boldsymbol{V}_{ui}$ computed from Step 2, we can construct the inequality system in Equation 5.4. To estimate the parameters $\boldsymbol{S}_u$ of the utility function $U(.)$, we follow the method in ([48], [29]), which is reported by the authors to be ''highly accurate, robust to noise and computationally efficient". We summarize the method as follows.

In practice, the choice set $\mathcal{P}$ may contain noise due to inconsistency or biases in consumer preferences, which make it impossible to find a perfect $\boldsymbol{S}_u$ satisfying all the inequalities in Equation 5.4. To cope with the problem, we introduce a positive slack variable ξ_l to each of the inequalities, as shown in Equation 5.11. The positive slacks play as the cost to violate any inequalities. Ideally, we would minimize these slack variables to reduce the number of violated inequalities. A natural way is to include the cost into the objective

function as shown in Equation 5.12.

$$
\begin{aligned}
U(u,i,price_i) &\geq U(u,j,price_j), \forall (i,j) \in \mathcal{P} \\
\sum_{k=1}^{K} s_k v_{uik} + s_{uprice} price_i &\geq \sum_{k=1}^{K} s_k v_{ujk} + s_{uprice} price_j - \xi_l, \forall (i,j) \in \mathcal{P} \\
\xi_l &\geq 0, l = 1 \dots |\mathcal{P}|
\end{aligned} \tag{5.11}
$$

A desired $\boldsymbol{S}_u$ is the one minimizing the number of violated inequalities in Equation 5.11, as well as all ξ_l. However, this leads to models that overfit to the current data, or even worse the noise and inconsistency in $\mathcal{P}$. As a consequence, the model is less accurate in capturing the actual consumer preferences. Hence, controlling the model complexity is crucial to estimate $\boldsymbol{S}_u$ from D. ([29]) shows a connection between controlling model complexity and minimizing $\|\boldsymbol{S}_u\|_2^2$, leading to the following optimization problem

$$
\min_{\boldsymbol{S}_u \in \mathbb{R}^K, s_{uprice}, \xi_l, \dots, \xi_{|\mathcal{P}|}} C \sum_{i=1}^{|\mathcal{P}|} \xi_i + \|\boldsymbol{S}_u\|_2^2 + s_{uprice}^2
$$

subject to

$$
\begin{aligned}
\sum_{k=1}^{K} s_{uk} v_{uik} + s_{uprice} price_i &\geq \sum_{k=1}^{K} s_{uk} v_{ujk} + s_{uprice} price_j - \xi_l, \ \forall (i,j) \in \mathcal{P} \\
\xi_l &\geq 0, l = 1 \dots |\mathcal{P}|
\end{aligned} \tag{5.12}
$$

In Equation 5.12, the objective is to minimize both the cost of violating the inequalities and the model complexity at the same time, meanwhile maintaining all the inequalities. Whether we emphasize on minimizing the cost or the model complexity reflects in the trade-off parameter C, which can be chosen through cross validation. Large C values mean that we penalize the cost of violation, while small C(s) allow a loose estimation of $\boldsymbol{S}_u$. Note that in some work, e.g. ([29]), C is the coefficient of the model parameters instead of the slack variables. In such cases, the role and effects of C still remains. As Equation 5.12 has the form of a Linear Support Vector Machine (SVM) [103],

and similar to the optimization problem in [48], we resort to the Cutting-Plane algorithm mentioned in [49] for estimating $\boldsymbol{S}_u$. The algorithm is the state-of-the-art method to solve linear SVM effectively and efficiently on large datasets.

Besides its performance, another advantage of the optimization problem above is its flexibility in incorporating domain knowledge into its constraint set to enforce specific output structures. We illustrate the advantage with the following two constraints.

- **Positivity constraint.** As mentioned above, consumers are assumed to lose some amount of utility by paying for the product at some prices. To incorporate this belief to our model, either we set $s_{uprice} \leq 0$ with $price > 0$, or $s_{uprice} \geq 0$ and $price < 0$. As both are equivalent in representing the negative utility of price, we consider the latter in our case. To ensure the positivity constraint of s_{uprice}, following [29], we add $s_{uprice} \geq 0$ to the constraint set in Equation 5.12. The inequality is equivalent to have a hypothetical consumer with preference over two hypothetical products with the corresponding diagonal matrices $\boldsymbol{V}_{ui} = \text{diag}(\{0, \ldots, 0, 1\})$ and $\boldsymbol{V}_{uj} = \text{diag}(\{0, \ldots, 0, 0\})$. Therefore, we can represent the inequality similarly to the other utility inequalities.

$$s_{uprice} \geq 1 - \xi_m \tag{5.13}$$

- **Anchor constraint.** Since the optimization problem in Equation 5.12 only seeks for $\boldsymbol{S}_u$ preserving the relative rankings in $\mathcal{P}$, the optimal $\boldsymbol{S}_u$ can lead to unrealistic values of willingness-to-pay, as long as the inequalities are satisfied. As we only observe the realization of consumers' actual willingness-to-pay, i.e., preference rankings among products, we make an assumption that consumers are willing to pay at most the maximum price in the category which the product in consideration belongs to.

This assumption can be realized through a constraint on the estimated willingness-to-pay or $\mathbf{v}_{ui}$. Similarly to the positivity constraint, we create another hypothetical consumer with preference over two hypothetical products with corresponding evaluation vector $\boldsymbol{V}_{ui} = \text{diag}(\{0, \ldots, 0, 0\})$ and $\boldsymbol{V}_{uj} = \text{diag}(\{v_{ui1}, \ldots, v_{uiK}, -\text{max\_price}\})$, which is equivalent to

$$0.0 \geq \sum_{k=1}^{K} s_{uk} v_{ujk} - s_{uprice} \times \text{max\_ price} + 1 - \xi_m \tag{5.14}$$

As violations in the constraint set are allowed at some cost, we need to set the cost of violating the two inequalities (Equation 5.13 and 5.14) high enough to enforce those two constraints. A natural way is to multiply both inequalities in the constraint set. Doing so on one hand magnifies the violation cost of that constraint, forcing it to be satisfied. On the other hand, those "dummy" constraints also affect the estimation quality of $\boldsymbol{S}_u$. To ensure that we add the minimal constraints, we include only the necessary number of constraints until the positivity constraints are satisfied.

For the remainder of this chapter, we learn w from the optimization problem in Equation 5.12 with both the positivity constraint (Equation 5.13) and anchor constraint set (Equation 5.14).

Training a personalized $\boldsymbol{S}_u$ on consumers with a few data points poses a challenge for a stable and valid model estimation. Following the idea of shared preferences, a possible remedy is to enforce the similarity of $\boldsymbol{S}_u$ to the ones belonged to her neighbors. The approach would increase the complexity of the Conjoint method as more constraints are introduced to the optimization problem. To refrain from complicating the problem unnecessarily, we propose a heuristic approach with the least modification to the original problem. Instead of training a personalized $\boldsymbol{S}_u$ for every consumer u, we consider a universal $\boldsymbol{S}$ applied for all consumers. The universal $\boldsymbol{S}$ is estimated by aggregating all

inequality systems (one for each consumer) into the constraint set in Equation 5.12. Note that no cross-consumer inequalities is created. Hence, $\boldsymbol{S}$ is still learned to fit each individual's preferences. Let us denote $\tilde{\boldsymbol{S}}$ the estimated universal model parameters from the aggregated inequality system.

5.2.4 Willingness-to-pay Extraction

Recall that we use five-fold cross-validation in Step 3 for selecting the best configuration for each sub-categegry. Given $\tilde{\boldsymbol{S}}$ from a fold in step 3, we can estimate both the product and price utility for all the consumer reviews. [46] suggests to transform utility into willingness-to-pay using the following equation

$$\text{wtp}_{ui} = \frac{\sum_k \tilde{s}_{uk} v_{uik}}{\tilde{s}_{price}} \tag{5.15}$$

5.3 Experiments

5.3.1 Experimental Setup

Data. As one of the largest online retailers in United States, Amazon has attracted a lot of visitors (around 140M unique visitors per month, starting from July, 2015 [2]). With such a large consumer base, the site is rich in consumer reviews and feedback on products. Hence, Amazon has become a familiar public data source for scholars to conduct studies related to consumer preferences (for e.g., [54], [8]).

To extract consumer willingness-to-pay we require both ratings and reviews. Among a plethora of categories on Amazon, Electronics is one of the most active categories which draw a lot of ratings and reviews from consumers. Since the category itself contains a lot of sub-categories, we sample a subset of popular product categories in which we are familiar with. To be specific, we collect review data from a sample of 6 sub-categories of Electronic prod-

[2] https://siteanalytics.compete.com/amazon.com/#.VwJV8mF96V4

Table 5.1: Review data statistics

	Number of customers	Number of products	Number of feedbacks	Rating distribution				
				1	2	3	4	5
Hard Drives	4,719	821	11,727	12%	5%	6%	21%	56%
Keyboards	3,954	2,061	8,851	8%	8%	11%	23%	50%
Mice	5,604	1,710	12,590	10%	8%	12%	20%	50%
Routers	3,894	801	8,844	20%	9%	9%	19%	43%
Speakers	2,734	906	6,159	8%	8%	11%	23%	50%
Tablets	4,442	1,413	10,052	15%	8%	11%	20%	46%

ucts, namely Computer Speakers (or Speakers in short), External Hard Drives (or Hard Drives in short), Keyboards, Mice, Routers and Tablets[3]. Each consumer feedback in the data contains the reviewer identity, product description, a rating score on a scale from 1 (dissatisfied) to 5 (satisfied), and review. Besides, each product may include additional information such as product title, price and categories. We cleanse the data by removing duplicate reviews (i.e., reviews of similar content), reviews with insufficient information (no title, or price or categories), reviews of consumers with only one feedback. Statistics of the preprocessed data are listed in Table 5.1. We also provide a detailed look into the selected products' price in Table 5.2 for its crucial role in the framework.

Table 5.2: Descriptive statistics on price information for each sub-category

	Price's statistics (dollar unit)			
	Min	Max	Mean	Standard Deviation
Hard Drives	3.95	999.00	118.10	122.68
Keyboards	0.01	389.00	36.56	42.11
Mice	0.87	474.06	28.83	40.57
Routers	3.28	684.00	82.74	93.15
Speakers	0.71	499.00	56.60	67.98
Tablets	0.56	999.99	146.95	174.91

Table 5.1 shows common trends in the rating distribution across subcategories. First, the average number of feedbacks for each consumer is less than 2.5, implying that most of the consumers only reviewed two products. This observation poses a challenge in performing Conjoint Analysis on individ-

[3]http://jmcauley.ucsd.edu/data/amazon/

uals due to their insufficient data, which motivates a *one-size-fits-all* Conjoint method in Section 5.2.3. Secondly, a majority of the ratings is inclined towards 4 and 5 stars, indicating that most of consumers are satisfied with their purchases. However, high ratings do not always accord with higher willingness-to-pay over prices as shown in Section 5.2.4. It is reasonable as ratings only indicate relative preference among products. Moreover, high ratings do not imply that reviewers like every aspect of the products. They may dislike a few features, which in turn affect to the overall utility.

For price information, an interesting observation in Table 5.2 is the minimum product price in every sub-category, which is as low as "giving-away" price. These may be prices on used products sold by other customers. Since there is no clear guidelines on a reasonable price range in a sub-category, we do not exclude those cases in our study.

5.3.2 Feature Identification

We apply the above process to identify product features in each sub-category above from its review data. For a sub-category, we first split each review into sentences and aggregate all into a sentence collection $\mathcal{D}$. Then, we perform hLDA on $\mathcal{D}$ to infer the hierarchical structure. As part of the input, we postulate a 3-level topic hierarchy, namely the root is for stop-words, the second layer contains topics related to product features and the last layer may contain noise or irrelevant topics. Product features are identified from the nodes at the second layer. We then determine product features in each sentence based on its associated topic path. Note that it is possible to have consumer review which does not address any product features. We filter out such reviews in this step. This general procedure is applied similarly on the six sub-categories.

We first create six sentence corpora $\mathcal{D}_{\text{Hard Drives}}$, $\mathcal{D}_{\text{Keyboards}}$, $\mathcal{D}_{\text{Mice}}$, $\mathcal{D}_{\text{Routers}}$, $\mathcal{D}_{\text{Speakers}}$ and $\mathcal{D}_{\text{Tablets}}$. Table 5.3 provides statistics for each corpus. We use

Sub-category	Number of sentences	Vocabulary size	Number of word tokens
Hard Drives	123,870	30,801	1,988,249
Keyboards	87,047	25,609	1,363,279
Mice	108,702	26,397	1,626,395
Routers	102,370	28,871	1,655,258
Speakers	66,106	22,635	1,036,453
Tablets	131,660	35,614	2,120,626

Table 5.3: Corpus statistics

the Gibbs sampling algorithm [4] to learn the model, with the following settings $\boldsymbol{\eta} = \{0.5, 0.01, 0.15\}$, $\boldsymbol{\gamma} = \{10.0, 0.1\}$, $m = 0.01, \pi = 1000$ on each corpus. As other Markov chain Monte Carlo algorithms, we iterate the algorithm many times until the model converges. For our data, we observe that after 10,000 samples the top frequent topics become stable. After the first 1000 iterations for burn-in, we collect samples for every 100 iteration. In total 91 samples are collected. Each sample contains a list of topics and their assignments to all the sentences. There are two separate tasks: 1) Identifying product features mentioned in each sentence corpus and 2) Identifying product feature mentioned in each sentence.

For the first task, as discussed above we seek for popular topics since they are more likely to mention product features. Such topics are stable with significant occurrences of representative words and frequently appear over samples. Hence, we select topics which occur over 91 samples. Product features are then identified from the selected topics based on the top frequent words in each topic. After filtering topics whose top-1 frequent word has less than 50 occurrences, Keyboards has **13** features, Hard Drives has **20** features, Mice has **16** features, Routers has **16** feature, Speakers has **13** features and Tablets has **21** features.

Due to space constraints, we only list top-5 frequently mentioned features for each sub-category in Table 5.4. All the identified features are essential parts

[4] http://www.cs.columbia.edu/∼blei/downloads/hlda-c.tgz

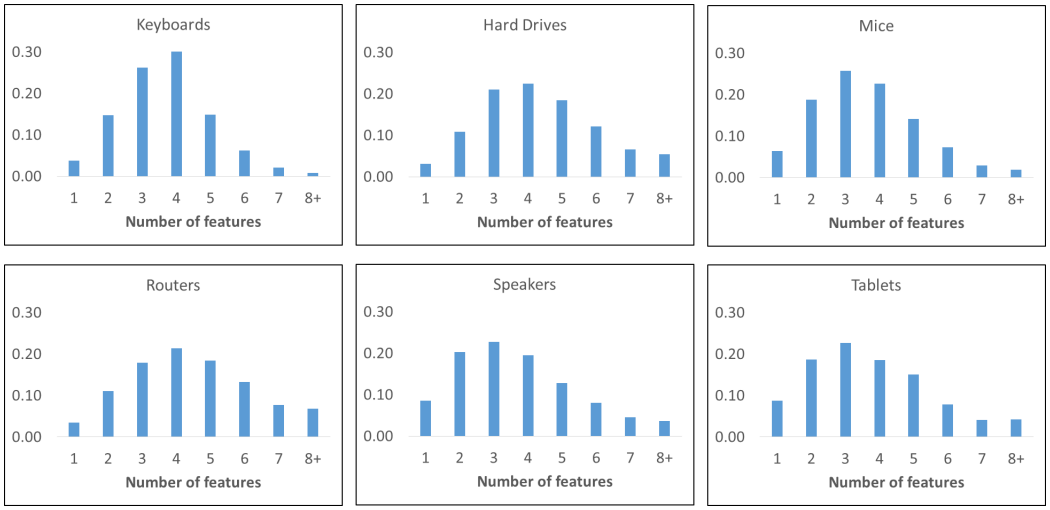


Figure 5.2: Distributions of number of features per review on each sub-category.

or functional features of products within the corresponding sub-category. For example, consumers concern about "keys" and "ergonomics" for keyboards, " sensitivity" and "battery" for mice, "power" and "I/O speed" for hard drives. Zooming in these features by examining the representative words provides interesting insights, such as consumers' feature usages. These insights are valuable for firms to improve not only their product design but also their business strategy. For example, consumers often mention popular streaming websites such as Nextflix, Rouku, etc. when they talk about the "Streaming ability" of routers. With this insight, a router company can design a product line specialized for streaming services, or it can collaborate with streaming websites to promote its products.

For the second task, to identify features mentioned in a given sentence, we track the most frequently assigned topics to the sentence over the selected samples. Product features are then identified from the frequently assigned topics based on the outputs of the first task. Table 5.5 shows the distribution of number of features mentioned in a sentence for each sub-category. A common observation across sub-category is that most of the sentences (98%) mention at most two features. We reason this observation is due to the sentence length.

Category	Feat. Id	Feat. label	Representative words
Hard Drives	6	WD Drives	digital, western, passport, essential
	10	Power	power, supply, cord, ac, adapter
	7	I/O Speed	mb, write, sec, read, sequential
	12	Interface	cards, sd, slot, pci, motherboard
	20	Partition	ntfs, partition, os, format, fat32
Keyboards	1	Keys	space, bar, alt, ctrl, pgup, del
	12	Media keys	volume, media, play, pause, mute
	11	Ergonomics	wrist, rest, palm, pain, strain, arm
	2	Mechanical keys	switches, cherry, mx, blue, red
	3	Sleeping mode	mode, sleep, wake, seconds
Mice	11	Receiver	usb, port, receiver, nano, plugged
	12	Sensitivity	dpi, speed, adjust, sensitivity
	14	Battery	aa, rechargeable, aaa, alkaline
	7	Mechanical mice	ball, clean, dust, dirt, roller
	9	Razer mice	razer, deathadder, diamondback
Routers	1	Privacy settings	password, admin, ssid, name, login
	2	Speed	mbps, 300, 150, 54, 100, 25, hmz
	14	Configurations	ip, address, dhcp, mac, static, dns
	8	Security	wpa, wep, wpa2, encryption
	13	Streaming ability	streaming, video, netflix, hd, tv
Speakers	10	Jack	jack, headphone, input, 3.5mm
	4	Internet radio	radio, internet, pandora, talk
	7	Size	inches, watt, tall, wide, dimensions
	8	Power	batteries, life, hours, charger
	13	O.S.	windows, xp, 7, vista, os, mac
Tablets	1	CPU	core, dual, quad, intel, atom
	3	Charging	usb, port, cable, charger, adapter
	14	Stylus pen	pen, notes, stylus, wacom, drawing
	16	External storage	card, sd, slot, micro, memory
	21	Multimedia	player, video, mp4, mp3, avi, mkv

Table 5.4: A sample of extracted product features (feat.) in each sub-category.

	Number of features per sentence		
Sub-category	1	2	≥3
Hard Drives	88.32%	10.02%	1.67%
Keyboards	88.10%	10.40%	1.50%
Mice	89.95%	8.84%	1.31%
Routers	88.86%	9.45%	1.69%
Speakers	90.42%	8.09%	1.49%
Tablets	89.87%	8.90%	1.23%

Table 5.5: Distribution of number of features mentioned in a sentence

Category	Example
Hard Drives	**Sentence #2188:** This is the third WD My Book drive that I've purchased and my experience with all three has been excellent. **Feature(s):** #6 (WD Drives) **Sentiment score:** 4 (positive)
Keyboards	**Sentence #38189:** The Cherry Blue keys are a perfect fit for me because I don't always push down so much when I type and I love the sound. **Feature(s):** #2 (mechanical keys) **Sentiment score:** 4 (positive)
Mice	**Sentence #177:** But I was tired to having to clean the ball every week, plus the local electronics store was running a nice deal on the IntelliMouse Explorer. **Feature(s):** # 7 (Mechanic mice) **Sentiment score:** 2 (negative)
Routers	**Sentence #101467:** It is easy to set up, has good range, and works very fast even when streaming video to multiple devices simultaneously. **Feature(s):** #13 (Streaming) **Sentiment score:** 5 (very positive)
Speakers	**Sentence #1681:** The only thing that I found was stupid was that it doesn't have a power button. **Feature(s):** #8 (Power) **Sentiment score:** 1 (very negative)
Tablets	**Sentence #16424:** Video playback is excellent. **Feature(s):** #21 (Multimedia) **Sentiment score:** 5 (very positive)

Table 5.6: Examples of review sentences mention product features and their overrall sentiment.

From Table 5.3, we can compute the average length of a sentence for all subcategories is around 15, which may be enough to describe two features. For a review, we aggregate the features mentioned in its sentences to obtain the feature list concerned by the reviewer. Figure 5.2 shows the histograms of number of features per review. We can see that for our data consumers are likely to comment on around 2-4 features in their reviews . Note that not all the sentences in a review would discuss a product feature. We illustrate this task on some sentences in Table 5.6.

5.3.3 Feature Evaluation

We experiment all the three measurements on the six sub-categories above. As there is no clear guide in literature to select a universal measurement, the most suitable method for each sub-category is then determined based on the quality of the corresponding estimated utility as shown in Step 3.

Example. Let consider an example from our data. The following is a 4-star review of a consumer with a consumer with ID code A1ZZ6VASR1H1YZ on a Cisco-Linksys router[5].

> *Good and easy setup for dynamic IP with Verizon DSL (4). Just want to let everyone that this machine works with both dynamic IP and fixed IP accounts with Verizon DSL in the NY/NJ area (4). With dynamic IP, I do encounter the occassional hiccup where I do have to disconnect the connection on the admin page and reconnect to re-establish connection (2). On a fixed IP account, the throughput was significantly higher and a lot smoother (for streaming audio/video) (2). Make sure to check back with Linksys for the constant firmware update (3).*

There are two features of the mentioned router identified by hLDA in the above review, namely the router's configurations and its streaming ability. To be specific, the first three sentences discuss about its configurations. The remaining two mention the router's streaming ability. We also put the corresponding sentiment score at the end of each sentence. Given the price of the product is \$74.99, the corresponding v_{uik}'s under each of measurement scheme are listed below:

- **Binary:** $\{(\text{“Configurations”} : 1); (\text{“Streaming”} : 1)\}$. Only two v_{uik} entries, representing “Configuration” and “Streaming ability” features are set to 1. The remaining entries are zero.

[5]http://www.amazon.com/dp/B00004SB92/

- **Frequency:** $\{(\text{“Configurations”} : 2); (\text{“}Streaming\text{”} : 3)\}$, indicating the router's “Configurations” are mentioned in the first three sentences and its “Streaming ability” are discussed in the remaining two.

- **Sentiment:** $\{(\text{“Configurations”} : 5); (\text{“}Streaming\text{”} : 10)\}$. Under this scheme, the “Configuration” entry equals to 5 for the 2 and 3 sentiment points of the two sentences addressing “Configuration” of the router. Similar explanation can be applied for the “Streaming ability” entry of 10.

5.3.4 Utility Estimation

As the two additional constraints would affect the estimation, we first train the optimization problem without adding them. Denote $\tilde{\boldsymbol{S}}$ as the optimal solution to the optimization problem. We evaluate $\tilde{\boldsymbol{S}}$ on its ability to predict unobserved relative rankings. A natural loss function is hence defined based on the portion of discordant pairs in testing data. This loss function is known as **pairwise loss** ([38]). The lower the **pairwise loss** is, the better $\tilde{\boldsymbol{S}}$ is.

To find the best $\tilde{\boldsymbol{S}}$ for each sub-category, we perform five-fold cross-validation with various trade-off cost parameters $\boldsymbol{C} = \{0.01, 0.1, 1, 10, 100, 1000\}$ and the three feature evaluation measures. There are total $5 \times 3 = 15$ configurations consisting of a specific C and a measure. We first split the consumers into five different groups with equal total number of pairs. For each configuration, we leave out one group for testing and optimize the model on the remaining four. We repeat the same process on each of the five folds, and report the average of **pairwise loss**. Table 5.7 shows the best average **pairwise loss** across different C values and a fixed evaluation scheme.

The best configuration varies among sub-categories, namely Hard Drives (Frequency, C = 1000), Keyboards (Sentiment, C=0.01), Mice (Sentiment, C = 0.01), Routers (Frequency, C = 1), Speakers (Sentiment, C = 0.01) and Tablets (Sentiment, C = 1) and are fixed in the remainder of the chapter.

Table 5.7: Pairwise loss on all six sub-categories (five-fold cross validation)

	Evaluation scheme		
Sub-category	Binary	Frequency	Sentiment
Hard Drives	0.224 (C=100)	**0.222** (C=1000)	0.227 (C=10)
Keyboards	0.272 (C=10)	0.276 (C=0.01)	**0.265** (C=0.01)
Mice	0.285 (C=0.01)	0.284 (C=0.01)	**0.277** (C=0.01)
Routers	0.319 (C=1000)	**0.304** (C=1)	0.319 (C=1)
Speakers	0.246 (C=1)	0.245 (C=0.01)	**0.243** (C=0.01)
Tablets	0.243 (C=1)	0.239 (C=10)	**0.235** (C=1)

Among the three measures, **Binary** has the worst performance. One possible explanation is that a two-outcome evaluation is not informative enough to capture the diversity in consumer preferences. On the contrary, the best evaluation measure is **Sentiment**. We postulate that the result is due to the expressiveness of **Sentiment** in representing consumer preferences. For the trade-off parameter C, small values (C=0.01 and C=1) show better performance than the large ones. Recall that small values of C would allow more violations in the constraint set which is mostly due to complication in learning a w which capture all the preferences. We reason this complication for training a universal w which aims to satisfy all the relative choices from different consumers.

With the best configuration for every sub-category data, we proceed to train the optimization problem with both positivity and anchor constraints. The quality of $\tilde{\boldsymbol{S}}$ is also measured through five-fold cross validation. Note that the dummy constraints are only involved in the training phase. We increase the number of dummy constraints by one until $\tilde{s}_{price} > 0$. The number of additional constraints in total for each sub-category and the corresponding **pairwise loss** is presented in Table 5.8. As expected, the **pairwise loss** increases with the presence of dummy constraints, ranging from 0% (Hard Drives) to 36% (Tablets). On average, each sub-category has a loss of 0.31 on test sets, which is still reasonable level of errors. Lower errors can be obtained if we focus only on the most confident cases, but with little data.

Table 5.8: Number of added constraints and the corresponding pairwise loss with and without adding dummy constraints (five-fold cross validation)

Sub-category	Number of added constraints	**pairwise loss**	
		Without constraints	With constraints
Hard Drives	0	0.222	0.222
Keyboards	134	0.265	0.329
Mice	74	0.277	0.303
Routers	44	0.304	0.313
Speakers	130	0.243	0.316
Tablets	204	0.235	0.320

5.3.5 Willingness-to-pay Extraction

Note that it is possible for the numerator in Equation 5.15 to get negative value (as we do not constrain other $\boldsymbol{S}_k$ ($k \leq K$) to be positive for more accurate estimation of $\boldsymbol{S}$). The corresponding willingness-to-pay in such case is also negative (due to $\boldsymbol{S}_{price} > 0$) and are discarded. From the remaining non-negative willingness-to-pay, we construct a $M \times N$ willingness-to-pay matrix W with M consumers and N products. Each positive entry W_{ij} represents the willingness-to-pay of consumer i on product j. In total we have five W matrices corresponding to each fold. We use the average matrix of the five in the remainder of this work. Figure 5.3 reports the distribution of the estimated willingness-to-pay of each sub-category. The estimated willingness-to-pay(s) vary within each sub-category. Besides, they also reflect the product prices. For example, keyboards and mice often have lower price than speakers or tablets. Some of the estimated willingness-to-pay are close zero, which is a reflection of the "give away" prices of used products as inputs.

More detailed statistics about each W can be found in Table 5.9. All six W matrices are sparse, with the percentage of non-zero entries ranging from 0.17% to 0.55%. As more reviews are observed, W would become denser. Moreover, on average, consumers are willing to pay less for hard drives, keyboards and mice, but more for routers, speakers and tablets. One explanation is that hard drives, keyboards and mice are replaced more often than the remaining cate-

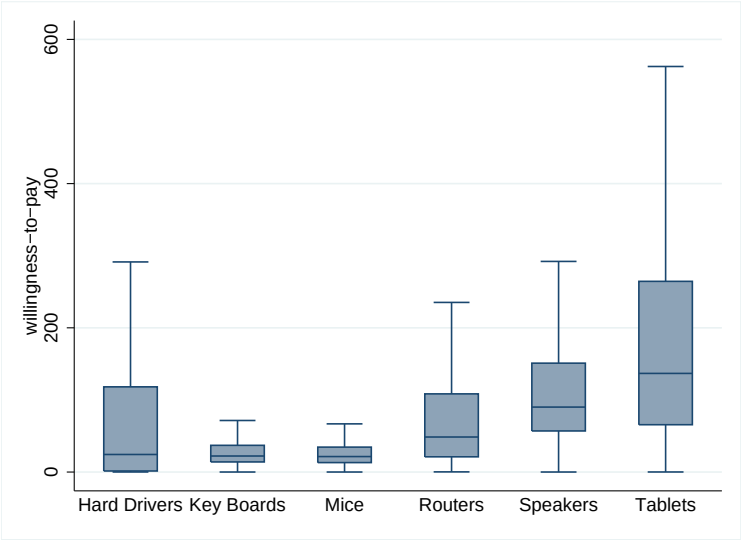


Figure 5.3: Distribution of the estimated willingness-to-pay

Sub-categry	Number of consumers	Number of products	Number of WTP	Average of WTP	Average of product price
Hard Drives	3,942	1,250	8,807	$30.03	$44.63
Keyboards	1,049	324	1,241	$87.31	$98.14
Mice	5,595	1,206	12,528	$27.38	$39.46
Routers	460	193	484	$80.96	$68.92
Speakers	2,712	595	6,096	$120.64	$74.62
Tablets	4,349	1,014	9,088	$218.01	$198.28

Table 5.9: Descriptive Statistics on estimated willingness-to-pay (WTP) on products of every sub-category.

gories, hence consumers would not appeal toward products with high prices. Meanwhile, routers, speakers and tablets are considered as long-term products, so consumers aim for highly-priced products with good quality.

5.4 Related Work

Recall from Section 1 that willingness-to-pay is the price where a consumer is indifferent in making purchase decision [46]. Knowing this quantity is essential not only to firms in design and pricing their products, but also to market researchers in studying market demand. Due to its practical and theoretical importance, measuring consumer willingness-to-pay has been drawn a lot of

attention in marketing research area, resulting in numerous approaches. Depending on the data sources, different measure approaches can be applied as suggested in [21]. On the high level, there are three common sources, namely market data, laboratory data and survey data. Reflecting their relative suitability in estimating willingness-to-pay at scale, we only brief each source below. For a complete analysis, interested readers can refer to [21, 82].

Market data or sales data is recorded from panel data and store scanner data respectively. Panel data is individual purchase data reported by a customer panel. Meanwhile, store scanner data is sales records from retail outlets. In general, market data is not suitable for estimating willingness-to-pay. Store scanner data is often collected at the aggregate level (e.g., store, time), hence contains few information on variance on product prices, or individual choice. Panel data does contain those information but entails high operation cost.

Laboratory data is collected from experiments conducted in laboratory settings or field settings. In the laboratory settings, consumers are given an amount of money to spend on a specific selection of goods. This set up would induce some biases in choice since the participants are already aware of the experimental situation (i.e., they do not use their own money to do "actual purchase" on goods). Field experiments, on the contrary, do not suffer from this problem. However, similar to panel data, this type of experiments also incurs considerably expenditures and amount of time intervals to track the responses to changes in price or product design. For both reasones, laboratory data is also not preferred to estimate willingness-to-pay in large scale.

The last data type is survey data. One survey type is to directly ask respondents to indicate their acceptable price range on some particular products. This method is prone to several drawbacks, mainly related to the accountability and trustworthiness of the response prices [21]. For example, consumers tend to state lower prices for their own benefits.

The other survey type indirectly measures willingness-to-pay based on re-

spondents' choices. Basically, the participants are asked to show their preferences on several product profiles with variance in features, for examples cameras with different colors or battery life. Their willingness-to-pay is estimated afterward using some models from economic theory of choice [46]. Whether the response data is a preference rating scores or the most appealing profile in sets of profiles, we have two types of approaches: rating-based (conjoint analysis) and choice-based conjoint (discrete choice analysis) method. The first approach focuses on estimation of willingness-to-pay at individual level, meanwhile the second targets the aggregate level estimation due to lesser observations on consumer choices. Each approach incurs different estimation methods. Both are shown to "have complementary strengths" and can be combined together [22].

Connection to our study. Our study is built on the rating-based Conjoint Analysis method for its applicability on estimating willingness-to-pay at large scale. To be specific, we apply the model addressed in [29] to estimate willingness-to-pay from online reviews. The main difference between our study and [29] is that rather than synthetic data with simulations on product features and their valuations from consumers, we conduct a full framework to extract those information from review data.

5.5 Summary

In this chapter we propose a framework to elicit consumer's willingness-to-pay from their online reviews. At the heart of the framework is Conjoint Analysis, a well-known method in the literature for the task. The framework consists of four steps. In the Step 1, product features mentioned in a review are identified. How consumer values each of the identified features is measured in Step 2. Given the outputs from Step 1 and 2, consumer's utility on the product mentioned in the review is estimated, and transformed to willingness-to-pay in Step 4.

Chapter 6

Finding Profit-Maximizing Bundling Configuration

6.1 Overview

Recall from Section 1.1.2 that *product bundling* is a practice of selling two or more items for one price. In general, there are three alternatives for *bundling*, namely *pure bundling*, *mixed bundling* and *unbundling* [1]. *Unbundling*, i.e., *pure components*, is the strategy in which firms price and offer their products separately, not as bundles. *Pure bundling* is the strategy in which firms sell only bundles, not their component products. *Mixed bundling* is the strategy in which both bundles and their components are released to the market. Which bundling strategy to choose and how to design bundles based on consumer demand are important issues for firms to achieve their business target, e.g. maximizing profit or maximizing consumer satisfaction.

Example. In Section 1.1.2, we show an example to illustrate *unbundling* and *pure bundling* with the corresponding profit of \$24.00 and \$26.00 respectively. Here we re-use the same example to illustrate the *mixed bundling* strategy in Table 6.1. Under the *mixed bundling* strategy, A, B and the bundle (A, B) are available to all the consumers at $\$8, \$11, \$12$ respectively. Since u_1

can get A at \$8, the bundle is appealing to her since it costs her only \$4 more to get B, which is equal to her willingness-to-pay for B. Similarly, u_3 also purchases the bundle as it would cost him \$1 to get A. u_2 would stick with A as she cannot afford the bundle. With \$1 of cost per a unit of A or B, the total profit would be $2 \times (\$12 - \$2) + \$8 - \$1 = \$27$, which is even higher than *pure bundling*.

Consumer	willingness-to-pay			Pure Bundling	Mixed Bundling		
	$w_{u,A}$	$w_{u,B}$	$w_{u,AB}$	$p_{AB} = \$15$	$p_A = \$8$	$p_B = \$11$	$p_{AB} = \$12$
u_1	\$11	\$4	\$15	✓			✓
u_2	\$8	\$2	\$10	✓	✓		
u_3	\$5	\$11	\$16				✓
Profit				*\$26*		*\$27*	

Table 6.1: Pure bundling v.s. Mixed Bundling

In this chapter we tackle the computational perspective in designing bundles. Assume that a firm[1] has an inventory of N products with corresponding consumer demand. How would it determine which products to sell as a bundle, and at what price to maximize their target? Answering these questions poses computational challenges as the number of possible bundles in consideration would grow exponentially relative to N (which is $2^N - 1$ in [36]). Moreover, since it is impractical to release all possible bundles to market, firm may seek for a subset of bundles instead. We call this the *bundle configuration* problem. The problem is even more computationally challenging as we have to consider various combinations of possible bundles. We provide a formal definition of this problem in below.

6.1.1 Problem Statement

Given willingness-to-pay as outputs from the **Preference Elicitation** framework, we now proceed to the main focus of our study, the *k-sized Bundling Configuration* problem which accepts a utility function mapping a bundle to a

[1]As our focus is on the computational aspect, for simplicity we assume a monopoly scenario in our study. Markets with competition is of our future direction.

real value and seeks for a set of non-overlapping bundles maximizing the total utilities. Forming non-overlapping bundles is a common scenario for businesses, for instances, a cable TV provider (e.g., Starhub, SingTel) may partition a large number of cable TV channels into a small number of non-overlapping bundles. Besides, non-overlapping bundles may reduce competitions among them when they are released in the same market.

Assumptions. We make the following assumptions about consumers' purchase behaviors, which are conventionally used in the bundling literature ([36, 10]).

- *Single Price:* Each unique bundle has one price, i.e., the seller demands the same price from consumers.
- *Single Unit:* Each consumer demands 0 or 1 unit of an item. This assumption can be relaxed if the number of units demanded is specified in the data.
- *No Budget Constraint:* Consumers are able to purchase any item, given sufficient willingness- to-pay.
- *No Supply Constraint:* Seller can sell to any number of consumers. For information goods, the marginal cost of providing an item to one more consumer is very low.
- *Strict additivity on willingness-to-pay:* Consumer's willingness-to-pay for a bundle is equal to the total willingness-to-pay for the component products.
- *Strict additivity on cost:* The variable cost of a bundle is equal to the total variable cost of each component.

In the bundling literature there are three types of willingness-to-pay structure for bundles, namely strict additivity, sub additivity and super additivity

([104]). Although we assume the strictly additivity structure in our study, the remaining two are also applicable to our problem. Similarly, our work is also flexible with various cost structures.

Utility-maximizing function. We approach the bundling problem from the firm's point of view. For them, both profit and consumers' satisfaction/awareness are essential to keep their businesses going. We postulate a connection of the latter to consumers' surplus. Indeed, anyone would satisfy products giving them more surplus, or more values than their expectations. As gaining more profits would lead to less surplus and vice versa, we represent the trade-off between the two quantities as the parameter $\alpha \in [0, 1]$ in the utility function $f : 2^N \to \mathbb{R}$

$$f \triangleq (1 - \alpha) \times \text{profit} + \alpha \times \text{surplus} \tag{6.1}$$

Given a bundle as input, f would seek for the optimal price maximizing the total utility contributed proportionately by both profit and surplus regards to α. More discussions on f can be found in Section 6.2.1. Assume the firm would choose a fixed α to represent their strategy, together with f, we can define the *k-sized bundle configuration problem* as follows.

Notation. Before dwelling on the formal definition of the problem, we first introduce a list of notations used through the rest of this chapter.

- A size-k bundle refers to a bundle b of exactly $|b| = k$ component items.
- A k-sized bundle refers to a bundle b of size $1 \leq |b| \leq k$.
- W is the willingness-to-pay matrix containing information about consumers' willingness-to-pay on component products. W is created from outputs of the Preference Elicitation framework.
- $\mathcal{L}$ is the compatible rule set. It consists of rules indicating whether any two products are compatible to be bundled. We assume the rule set is

provided as a priori. A bundle is said to be compatible if any two of its components are compatible regarding to $\mathcal{L}$.

Problem Formulation. There are two instances corresponding to pure bundling and mixed bundling, namely *k-sized Pure Bundling* and *k-sized Mixed Bundling*.

Problem 1 (k-sized Pure Bundling)**.** *Given W, f, $\mathcal{L}$, and an integer $k \geq 1$, find the* bundle configuration $\mathcal{X}_I$*, containing k-sized bundles meeting the following conditions:*

1. $\bigcup \mathcal{X}_I = I$*, i.e., the union of sets in $\mathcal{X}_I$ is I*
2. $\forall b_1, b_2 \in \mathcal{X}_I$*,* $b_1 \cap b_2 \neq \emptyset$ *implies* $b_1 = b_2$
3. $\forall b \in \mathcal{X}_I$*,* b *is a compatible bundle according to* $\mathcal{L}$*.*
4. $\sum_{b \in \mathcal{X}_I} f(b)$ *is maximized, i.e., there is no other partitioning of I with a higher overall utility.*

$$\mathcal{X}_I = \underset{\mathcal{X} \text{ is a configuration of } I}{\operatorname{argmax}} \sum_{b \in \mathcal{X}} f(b) \tag{6.2}$$

The bundle configuration that meets the above conditions is called *optimal*. The parameter k limits the maximum size of the bundles. For information goods (e.g., cable television), bundle sizes can grow very large, e.g., hundreds in ([10]). For physical goods (e.g., books), smaller bundle sizes may be more appropriate. The first condition specifies how the collection of all bundles should make up the full set of items. The second condition characterizes the pure bundling strategy, by requiring that $\mathcal{X}_I$ is a strict partition of I, i.e., no overlap between bundles. This prevents having both a bundle, as well as its component items both available. As discussed above, we consider a business scenario of selling non-overlapping bundles. There may be other scenarios requiring overlapping bundles, and such scenarios are out of the scope of the

current work. Finally, the last condition specifies the utility maximization objective.

Mixed bundling has the distinction of allowing both a bundle and its component items being available. This is done by the subsumptions in the second condition in Problem 2.

Problem 2 (k-sized Mixed Bundling)**.** *Given W, f, $\mathcal{L}$ and an integer $k \geq 1$, find the* bundle configuration $\mathcal{X}_I$*, containing k-sized bundles meeting the following conditions:*

1. $\bigcup \mathcal{X}_I = I$*, i.e., the union of sets in $\mathcal{X}_I$ is I*
2. $\forall b_1, b_2 \in \mathcal{X}_I$*, $b_1 \cap b_2 \neq \emptyset$ implies $b_1 \subseteq b_2$ or $b_2 \subseteq b_1$*
3. $\forall b \in \mathcal{X}_I$*, b is a compatible bundle according to $\mathcal{L}$.*
4. $\sum_{b \in \mathcal{X}_I} f(b)$ *is maximized, i.e., there is no other partitioning of I with a higher overall utility.*

$$\mathcal{X}_I = \underset{\mathcal{X} \text{ is a configuration of } I}{\operatorname{argmax}} \sum_{b \in \mathcal{X}} f(b) \tag{6.3}$$

Example. As neither bundling form nor cost structure affects to the number of possible bundles, we slightly modify the example in Section 1.1.2 to illustrate the *k-sized pure bundling* problem. First we consider zero-cost products. Second, we introduce an addition product C. The new willingness-to-pay matrix W which contains all the customers' willingness to pay on every product and possible bundles as shown Table 6.2,

	Components			Bundles			
Consumer	(A)	(B)	(C)	(A, B)	(A, C)	(B, C)	(A, B, C)
u_1	\$11.00	\$4.00	\$3.00	\$15.00	\$14.00	\$7.00	\$18.00
u_2	\$8.00	\$2.00	\$7.00	\$10.00	\$15.00	\$9.00	\$17.00
u_3	\$5.00	\$11.00	\$6.00	\$16.00	\$11.00	\$17.00	\$22.00

Table 6.2: The willingness-to-pay matrix W

# Configuration		Price	Consumers u_1	u_2	u_3	Profit	Total profit
1	(A)	\$8.00	✓	✓		\$16.00	
	(B)	\$11.00			✓	\$11.00	\$39.00
	(C)	\$6.00		✓	✓	\$12.00	
2	(A, B)	\$15.00	✓		✓	\$30.00	
	(C)	\$6.00		✓	✓	\$12.00	\$42.00
3	(A, C)	\$11.00	✓	✓	✓	\$33.00	
	(B)	\$11.00			✓	\$11.00	\$44.00
4	(A)	\$8.00	✓	✓		\$16.00	
	(B, C)	\$7.00	✓	✓	✓	\$21.00	\$37.00
5	(A, B, C)	\$17.00	✓	✓	✓	\$51.00	\$51.00

Table 6.3: All the *Bundling configurations* with corresponding profit.

As shown in Table 6.3, there are five *Bundling configuration*s the seller can consider in the above example. In the *Configuration #1*, A, B and C are offered separately, which is the *Unbundling* strategy. There are two consumers who would purchase product A at \$8, yielding a revenue (or profit due to zero-cost products) of \$16. Similarly we have \$11 and \$12 in profit for selling component B and C respectively. Totally, *Configuration #1* yields a profit of \$39. The same reasoning applies for the rest of configurations. The *Configuration #5* selling a bundle of three products returns the highest profit at \$51.

Even though there are at most N bundles in every *bundling configuration*, we still need to enumerate all possible bundles. Plus, as a *bundling configuration* forms a partition of the given product set, the number of *bundling configuration* in consideration for N products is the Bell number [2], defined as follows

$$B_0 = 1, B_1 = 1, \ldots, B_N = \sum_{k=0}^{n-1} \binom{N-1}{k} B_k \tag{6.4}$$

[12] shows that the asymptotic upper-bound of B_N is $(\frac{0.792N}{ln(N+1)})^N$, which again grows exponentially in relative to N.

[2] https://en.wikipedia.org/wiki/Bell_number

6.2 Utility Maximizing Objective

6.2.1 Firm's Utility Objective Function

As mentioned in Section 6.1.1, we look at the problem of a firm trying to maximize its utility, be it profits or consumer surplus. We also formulate the utility function in Equation 6.1, in which the trade-off between profit and consumer surplus is captured through the parameter α. We set $\alpha = 0$ to represent a profit-maximization goal, and $\alpha = 1$ to maximize consumers' surplus. Given the willingness-to-pay matrix W for each sub-category, how would firm price products to maximize its utility ? We discuss this research question for component products in the following section and for bundled products in the following Section 6.3.1.

6.2.2 Utility-Maximizing Prices for Single Products

Profit of a product depends on how firm sets its price. For a given price, the profit is defined to be the remaining amount of revenue after deducting costs. While there are two types of costs (fixed cost and variable cost), only variable cost is affected by the quantity sold (which in turn is affected by pricing). Given the willingness-to-pay matrix W containing the maximum amount of money consumer i is willing to pay for product j, we can formulate the profit function for a product j at price p as follows

$$\begin{aligned} \text{profit}(j,p) &= \ quantity\_sold \times (price - variable\_cost) \\ &= \sum_{\text{for each consumer } i} \mathbb{I}(W_{ij} \geq p)(p - cost(j)), \end{aligned} \tag{6.5}$$

where the indicator function $\mathbb{I}(.)$ returns 1 when the argument is true, otherwise 0. Recall that W can be obtained from online reviews as described in Chapter 5.

As mentioned in the *Single Unit* assumption in Section 6.1.1, consumer

requires one unit for any product. She would make a purchase of that product when her respective willingness-to-pay is higher than its price. The quantity sold is therefore the number of consumers purchasing the item, indicated by the summation of indicator function over consumers. When the variable cost is zero or very small, such as in certain markets like digital goods (e.g., cable TV, video on demand) [10], profit is equivalent to revenue. For physical goods, the cost often accounts up to some percentage in the product's market price. In our case of Computer Peripheral products, we follow [25] to set the cost function $cost(j)$ equal to 70% of the corresponding listed price on Amazon.

By definition, consumer earns surplus from his/her every purchase, which also depends on product price. The total surplus firm can "obtain" by setting the product price at p

$$\text{surplus}(j,p) = \sum_{\text{for each consumer } i} \mathbb{I}(W_{ij} \geq p)(W_{ij} - p) \tag{6.6}$$

From Equation 6.5 and Equation 6.6, we can formulate Equation 6.1 as a function of price

$$f(j,p) = \sum_{\text{for each consumer } i} \mathbb{I}(W_{ij} \geq p)((1-\alpha)\times(p-cost(j))+\alpha\times(W_{ij}-p)) \tag{6.7}$$

We can compute the maximum utility firm can get from selling product j

$$f(j) = f(j,p^*) = \max_p f(j,p), \tag{6.8}$$

with $p^* = \arg\max_p f(j,p)$.

Searching for p^* involves investigating each candidate price p. In real-life scenarios, the seller would have a price list of T price levels. At worst, the price levels are the smallest atomic unit (e.g., cents). More commonly, it varies by larger increments.

Given such a price list, we associate each price level with a bucket and

place each of the M consumers into a bucket according to her willingness-to-pay W_{ij}. Identifying the bucket can be done through hashing (if equi-distanced price levels), or binary search (if arbitrary price levels). The optimal price level is determined by iterating through the T buckets. The number of buckets is usually fixed as a constant, and does not grow with the problem size. For experiments, we use 100 buckets, as we show later that larger numbers of buckets do not yield much profit gain. The complexity of pricing is therefore consumer-dependent at $O(M)$.

In the Section 6.4.2, we experiment the optimality of $f(j)$ on various settings of α and T with the estimated W from the previous chapter.

6.3 Bundling Configuration

6.3.1 Utility-Maximizing Prices for Bundles

Product bundles, or packages, are commonly defined to be a set of products. The matrix W from Section 5.2.4 only specifies the willingness-to-pay for single products. Suppose we define a bundle $b \subseteq I$ as a set of items. According to the assumption of Strict additivity on willingness-to-pay (Section 6.1.1), a consumer i's willingness-to-pay for a bundle b, denoted $\mathcal{W}_{i,b}$, can be obtained as follows

$$\mathcal{W}_{i,b} = \sum_{j \in b} W_{ij} \tag{6.9}$$

Given any bundle b and its corresponding consumers' willingness-to-pay $\mathcal{W}_{\cdot,b}$, we can similary compute the maximum utility firm can get from b following the Equation 6.8.

$$\begin{aligned} f(b) = f(b, p^*) &= \max_p f(b, p) \\ &= \max_p \sum_{\text{for each consumer } i} \mathbb{I}(\mathcal{W}_{i,b} \geq p)((1-\alpha) \times (p - cost(b)) + \alpha \times (\mathcal{W}_{i,b} - p)) \end{aligned} \tag{6.10}$$

Note that the bundle b is defined to be a group of existing products, producing an additional unit of b incurs a total cost to product an extra unit of each component, i.e. $cost(b) = \sum_{j \in b} cost(j)$ (i.e., the assumption of Strict additivity on cost in Section 6.1.1)

6.3.1.1 Pure Bundling v.s. Mixed Bundling

We apply the same price search strategy in Section 6.2.2 to find p^* in Equation 6.10. The strategy is slightly different regarding to the bundling form of b. For *pure bundling*, only a bundle or its components are on offer. For *mixed bundling*, a consumer's response to a bundle depends not only on the bundle's price, but also on the prices of its components. For example, i may be willing to pay $W_{iA} = \$11$ and $W_{iB} = \$4$. Correspondingly, we have $\mathcal{W}_{i,AB} = \$15$. Suppose that mixed bundling offers the following prices: $p_A = \$8$, $p_B = \$8$, and $p_{AB} = \$15$. One may think i would purchase the bundle, because $\mathcal{W}_{i,AB} \geq p_{AB}$. That would be a counter-intuitive outcome from ([1]). i could purchase A alone for \$8. To "upgrade" from A to the bundle $\{A, B\}$ would incur an implicit price for B of $p_{AB} - p_A = \$7$, more than i's willingness-to-pay for B (\$4). Therefore, i would purchase item A alone. In an alternative scenario where the offer is: $p_A = \$11$, $p_B = \$4$, and $p_{AB} = \$14$, i would purchase the bundle instead.

In other words, for pure bundling, the adoption decision is determined based on whether $p_{AB} \leq \mathcal{W}_{i,AB}$, which means that the pricing of a bundle and its components can be done independently. In contrast, for mixed bundling, it is based on whether $p_{AB} - p_A \leq W_{iB}$ and $p_{AB} - p_B \leq W_{iA}$. This induces dependencies between prices of a bundle (p_{AB}) and its components (p_A and p_B). In most application scenarios, we expect that components are by default on offer, and the seller seeks greater utility through bundling. Therefore, for mixed bundling, we adopt an *incremental* policy where the prices of components are determined first, and the price of a bundle is conditioned on the prices of

components. We would investigate a relaxation of this policy as future work.

We apply the usual constraints for mixed bundling ([35]) to ensure a bundle would be a viable alternative to its components. First, $\forall i \in b,\ p_b > p_i$, i.e., the bundle price must be higher than any of its component's. Second, $p_b < \sum_{i \in b}\ p_i$, i.e., the bundle price is lower than the sum of its components' prices. These are not applicable to pure bundling.

6.3.1.2 Compatible Bundles

In practice, not all combinations of products can be realized into bundles. Some products may not be bundled together for many reasons such as irrelevance in usage (e.g. a bundle of tooth brush and hard drive), negative effects (e.g. toilet papers and cook books). We call these products are not compatible to each others. Since firm may have its own rule set to define compatibility between product, we model firm's compatible rule set as follows.

Assume that firm's compatible rule set only defines on two products, let denote $\mathcal{L} = \{(j_x, j_y)_l\}$ the compatible rule set. Each product pair (j_x, j_y) is said to be compatible. For pairs of two identical products, it depends on whether firm includes such pairs in $\mathcal{L}$. Note that this representation of rule set is also applicable to rule sets involving more than two products. Based on $\mathcal{L}$, we can define a compatible bundle transitively as a group of products whose any of its subset, excluding itself and empty set, is compatible. For example, a bundle of size 3 is compatible if it does not have any incompatible 2-sized subsets. Similarly, a bundle of size 4 is said to be compatible if it does not have any incompatible 2-sized and 3-sized subsets. In the following sections, unless being stated explicitly, we assume $\mathcal{L}$ contains all pairs of products.

In the following, we first address the 2-sized bundle configuration problem (Section 6.1.1), before addressing the case of $k \geq 3$.

6.3.2 Optimal Solution for 2-sized Bundling

For $k = 2$, the bundle configuration may contain bundles of size 1 or 2. There are N candidates of the former, and $N(N-1)/2$ candidates of the latter. Out of these candidate bundles, we need to select a subset of them to make up the bundle configuration $\mathcal{X}_I$ with the highest utility.

First, we describe the *pure bundling* strategy. Bundles in $\mathcal{X}_I$ do not overlap with each other (see Section 6.1.1). If we model each item as a node in a graph (i.e., each bundle of two as an edge between two nodes, and each bundle of one as an edge from a node to itself), a valid bundle configuration is a set of edges, such that no two edges are incident on the same node and these edges collectively need to cover all the nodes, i.e., non-overlapping and no overage. In graph theory, such a collection of edges are known as a graph matching.

2-sized bundling can thus be reduced to graph matching. We construct a "complete" graph $G(V, E)$, where each vertex in V corresponds to an item in I, i.e., $|V| = N$. The set of edges E contains $N + N(N-1)/2$ edges. There is one edge from a vertex to itself (a candidate bundle of size 1). There is also an edge between any pair of vertices (a candidate bundle of size 2). Every edge is weighted by the maximum utility of the item or bundle (see Section 6.2.2 and Section 6.3.1). It is easy to see that the optimal 2-sized bundling configuration is a maximum weight matching in G, with the highest sum of edge weights (total utility). The advantage of this formulation is the existence of polynomial time algorithms for maximum weight matching, e.g., the Edmonds algorithm [28], with a complexity of $O(|E||V|^{\frac{1}{2}})$. For experiments, we use the LEMON library[3]. To compute the edge weights (see Equation 6.8 and Equation 6.10), we need to scan the M users once for each edge, which takes $O(MN^2)$. The complexity of this algorithm is thus $O(MN^2 + N^{2.5})$.

For *mixed bundling*, a similar formulation applies, with an adjustment whereby the edge weight between two different vertices is the maximum util-

[3]http://lemon.cs.elte.hu/trac/lemon

ity expected from offering both a bundle and its two components (see Section 6.3.1). A bundle is feasible if offering both the bundle and its components bring in more utility for firm than offering its components alone.

6.3.3 Complexity of k-sized Bundling

Complexity. The case for $k \geq 3$ is more complex. Finding the optimal selection of potential bundles is intractable.

Theorem 1. *3-sized pure bundling problem is NP-hard.*

Proof. Sketch proof.

3-sized pure bundling problem can be expressed in terms of finding a maximum matching in a hypergraph containing edges of size-1, size-2, and size-3. We now show that 3-sized pure bundling problem is NP-hard through a reduction from the maximum 3-uniform hypergraph matching (known to be NP-hard [42]).

For any instance of maximum 3-uniform hypergraph matching problem for a hypergraph H, we can construct an instance of maximum hypergraph matching problem for H', where H' is not necessarily 3-uniform. H' is created as follows. For each original edge in H, we create an edge in H' with the weight $3 + \Delta$, where Δ is a fixed positive constant. In addition, we add "dummy" edges of size-1 (weight 1), size-2 (weight 2), and size-3 (weight 3) to H' that are not already in H. S' is a maximum matching in H', if and only if S is a maximum matching in H. Since finding the maximum matching S in H is NP-hard, finding the solution S' in H' (the 3-sized pure bundling problem) is also NP-hard. □

Because pure bundling is a special case of mixed bundling where a bundle and its components are not allowed to co-exist, mixed bundling is likely as complex as, if not more complex than pure bundling, because a mixed bundling configuration may contain both a bundle and its components.

6.3.4 Optimal Solution for k-sized Bundling

If we first enumerate all the $K = 2^N - 1$ potential bundles, the problem of pure bundling configuration among these potential bundles can be reduced to an instance of weighted set packing [24], for which there exists a solution with a known approximation bound. In weighted set packing, the input consists of K sets, where each set b_j has a known weight w_j. The objective is to find a collection of sets (that are pairwise disjoint) resulting in the highest aggregate weight. In our case, a set is a candidate bundle, and its weight is its maximum utility.

Optimal Solution. The optimal solution to *weighted set packing*, and thus to pure bundling problem, can be computed using the following Integer Linear Program (ILP). For every potential bundle b_j, we associate it with a binary variable x_j, which takes the value of 1 if b_j is selected as part of the solution, and 0 otherwise. Each b_j is also associated with a real-valued positive weight w_j, which represents its maximum utility. The objective is to determine x_j's to maximize the utility using a combination of non-overlapping bundles.

$$\begin{aligned} \text{maximize} \sum_{j=1}^{K} x_j \times w_j \\ \text{subject to} \sum_{b_j:\ i \in b_j} x_j \leq 1, \text{for all } i = 1, \ldots, N \\ x_j = \{0, 1\}, \text{for all } j = 1, \ldots, K \end{aligned}$$

This program is intractable for large N, which generates $K = 2^N - 1$ candidate bundles. In turn, the ILP has to find a solution within a space of $2^K - 1$ possible settings of x_j's. In practice, it is computable only for very small problem sizes. For experiments, we use the Gurobi ILP solver [4].

[4] http://www.gurobi.com/

6.3.5 Heuristic Solutions for k-sized Bundling

Here, we propose three heuristic algorithms for $k \geq 3$. Both algorithms can be applied to pure and mixed bundling. For clarity, we would first describe pure bundling, and highlight mixed bundling's differences in Section 6.3.5.4.

6.3.5.1 Approximable Solution to Weighted Set Packing

In Section 6.3.4, we show an optimal solution to our *Bundling configuration* problem by casting it as an instance of the *weighted set packing* problem. Since weighted set packing itself is also NP-hard, there exists approximation algorithms. The current best known solution is a greedy approach that repeatedly selects the next set with the highest average weight per item and removes other sets that overlap with the selected sets from future consideration. For N items, this approach is guaranteed to produce a solution within a factor of $\sqrt{N}$ less than the optimal solution in [33].

Though this connection to weighted set packing allows us to establish the approximability of the problem, in practice existing weighted set packing solutions are still intractable for our scenario due to the requirement of enumerating and computing the utility values of all possible candidate bundles beforehand, a step that by itself has an $O(M \cdot 2^N)$ complexity. This is in addition to the complexity of weighted set packing algorithm (ILP or greedy). Therefore, we develop more efficient heuristic algorithms below, which do not require prior enumeration of all subsets. We will compare them experimentally to the weighted set packing solutions in Section 6.4.4.1.

6.3.5.2 Matching-based Algorithm

Since the problem is tractable for $k = 2$ but not for $k \geq 3$, instead of solving it directly for k-sized bundles, one promising approach is to iteratively construct ever larger bundles. The pseudocode of this matching-based algorithm is given in Algorithm 1. We describe the *pure bundling* strategy here. For now, please

ignore the lines specific to mixed bundling.

Each iteration is concerned with finding the best bundle configuration from the current components. The same Edmonds algorithm can be used in each iteration. For example, in the first iteration, we form size-2 bundles from combinations of size-1 bundles. In the second iteration, we treat these size-2 bundles as if they are singular items, and create another instance of 2-bundling problem with the size-1 and size-2 bundles as initial bundles, allowing a size-3 bundle to be created from a combination of a size-1 bundle and a size-2 bundle, or a size-4 bundle to be created from a combination of two size-2 bundles. This continues till we reach the maximum size k, or until there is no more gain in utility.

We apply two pruning strategies to improve efficiency. In the first iteration, instead of all possible size-2 bundles, we only consider pairs of items for which at least one customer has non-zero willingness to pay for both. We cannot extract any positive remaining willingness to pay for the second item from customers who each want to buy only one item.

In subsequent iterations, when forming the edges in the graph, other than self-loop edges (the status quo), the other pruning strategy is to only introduce a new edge involving at least one newly-formed vertex in the current iteration. Edges in previous iterations that do not form a collapsible vertex will never form a bundle in the subsequent iterations as they are not favored over their components. Because of this "diminishing" effect of bundling, the number of iterations will effectively be bounded, as analyzed below.

At the start, there are N vertices and in the order of N^2 edges. For any given iteration, an edge between two vertices is either merged (if selected in the matching) or deleted. In the worst case, all non self-loop edges will be selected by the matching. Thus, the maximum number of edges, $|E|$, is no more than $(\frac{N}{2})^2$ after one iteration, $(\frac{N}{4})^2$ after two iterations, and so on. Taking into account the Edmonds algorithm's complexity of $O(|E||V|^{\frac{1}{2}})$, this is a geometric

Algorithm 1 Matching-based Algorithm

```
Initialize X_I to be a set of size-1 bundles.
Initialize X'_I to be an empty set.
Initialize R with the revenue of components.
while true do
    Construct a graph G with X_I as vertices.
    Populate G with edges involving newly-formed bundles.
    Compute the weight of each edge (see Eq. 6.8 and Eq. 6.10).
    Obtain the maximum weight matching S in G.
    Compute R', the weight or revenue of S.
    if R' ≤ R then
        Break.
    end if
    R ← R'
    for each selected edge in S do
        Remove the edge's vertices from X_I.
        Collapse the edge into a new vertex in X_I.
        if mixed bundling then
            Insert the edge's vertices into X'_I.
        end if
    end for
end while
Return X_I ∪ X'_I.
```

series of the form $a + ar + ar^2 + ...$, with $a = N^{2.5}$ and $r = \frac{1}{2}^{2.5}$, whose summation is bounded by $\frac{a}{1-r} = \frac{N^{2.5}}{1-\frac{1}{2}^{2.5}}$. Since the denominator is a constant, the complexity of matching across iterations is effectively still $O(N^{2.5})$.

To compute the edge weights, we need to scan the database of M users' willingness to pay once in every iteration, and update the utility computation of all newly-formed bundles. Because the number of edges $|E|$ diminishes in a geometric series as above, the utility computation requires $O(MN^2)$. The complexity of this algorithm is $O(MN^2 + N^{2.5})$. Realistically, the number of items N to be bundled is not extremely large. The number of users M may in some cases be large but it may be sufficient to take a significant sample of users, rather than using the data of all users.

6.3.5.3 Greedy Algorithm

The above matching-based approach is oriented towards finding the best configuration globally across all bundles in the partition $\mathcal{X}_I$. An alternative ap-

Algorithm 2 Greedy Algorithm

Initialize $\mathcal{X}_I$ to be a set of size-1 bundles.
Initialize $\mathcal{X}'_I$ to be an empty set.
Initialize R with the revenue of components.
while true **do**
  for every pair of elements $b_1, b_2 \in \mathcal{X}_I$ **do**
    Form a candidate bundle $b' = b_1 \cup b_2$ of size $\leq k$.
    Compute the absolute gain in revenue: $r_\Delta = r_{b'} - r_{b_1} - r_{b_2}$ (see Eq. 6.8 and Eq. 6.10).
  end for
  Let $b' = b_1 \cup b_2$ be the candidate bundle with highest absolute revenue gain r_Δ.
  if $r_\Delta \leq 0$ **then**
    Break.
  end if
  $\mathcal{X}_I \leftarrow \mathcal{X}_I - \{b_1, b_2\}$
  $\mathcal{X}_I \leftarrow \mathcal{X}_I \cup b'$
  $R \leftarrow R + r_\Delta$
  if mixed bundling **then**
    $\mathcal{X}'_I \leftarrow \mathcal{X}'_I \cup \{b_1, b_2\}$
  end if
end while
Return $\mathcal{X}_I \cup \mathcal{X}'_I$.

proach is to find only *one* best new bundle in each iteration. This new bundle can then immediately participate in the selection of the next bundle.

Algorithm 2 encapsulates this approach. In each iteration, it tries to perform a merging operation involving two existing bundles that result in the highest absolute gain in utility over the component bundles. This newly merged bundle then participates in the next iteration searching for the next best merged bundle. This continues until a stopping condition is met. One natural stopping condition, which we adopt in this paper, is when there is no more utility gain. An alternative stopping condition is to continue anyway till there is only a single bundle of N items, and then traversing all previous solutions to find the one with the maximum utility. Empirically, this would increase running time significantly without producing meaningful utility gain.

The algorithm may take up to N outer iterations, because in each iteration, the number of bundles in the configuration reduces by exactly 1, by collapsing two existing bundles into a new bundle. The first iteration involves $O(N^2)$

utility computations, as the utility values of all candidate bundles of size-1 and size-2 need to be computed. In each subsequent iteration, we only need up to N utility computations, involving the new bundle with an existing bundle. Each utility computation itself will need $O(M)$ (see Section 6.2.2). There is also the cost of picking the bundle with the maximum gain in each iteration, which differs in complexity depending on the specific implementation (e.g., priority queue involves $O(\log N)$ per insertion/removal). Therefore, the overall complexity is approximately $O(MN^2 + N^2 \log N))$.

6.3.5.4 Pure Bundling vs. Mixed Bundling

The above matching-based and greedy algorithms are applicable for both bundling strategies (pure and mixed), with several differences. For one thing, the key difference between the two is how the utility of a bundle is computed (see Section 6.3.1.1). Beyond that, another difference, as shown in Algorithm 1 and Algorithm 2, is that for mixed bundling, we retain in $\mathcal{X}'_I$ the subset of bundles replaced in previous iterations. These represent components that are also available to consumers, in addition to the subsuming bundles, which are finally returned as outputs of the algorithms. In contrast, for pure bundling, $\mathcal{X}'_I$ remains an empty set throughout.

6.4 Experiments

In this section, we first examine the profitability of the proposed algorithms under various settings, and compare it with other existing methods (Section 6.4.3). Moreover, we also examine the efficiency of the proposed methods in Section 6.4.4.

6.4.1 Experimental Setup

Datasets. We conduct experiments with two types of willingness-to-pay data. The first one consists of the willingness-to-pay matrices as described in Chapter 5. There are six of them, each corresponds to a sub-category. The willingness-to-pay matrix for Mice has the most number of consumers ($\sim$ 5,600 consumers). Routers has the least of 460 consumers. The sparsity percentage of the willingness-to-pay matrix for each sub-category is 0.18% on Hard Drives, 0.37% on Keyboards, 0.19% on Mice, 0.55% on Routers, 0.38% on Speakers and 0.21% on Tablets. The second type of willingness-to-pay data is to convert from rating, as mentioned in Section 5.1.1. Since similar observations can be drawn from both types of willingness-to-pay data, as shown in Figure 6.3a and Figure 6.5 for mixed bundling, and Figure 6.3b and Figure 6.6 for pure bundling, in this section we focus on the review-based one.

6.4.2 Utility-Maximizing Prices for Single Products

Varying T. We experiment the effect of various T on the firm's utility. Figure 6.1 shows the results for each sub-category with $\alpha = 0$. We omit figures of other α due to similar trends. The horizontal axis represents different number of price buckets. The vertical axis describes the profit firm can earn. All sub-categories have similar trend. As the price gets finer (i.e., more buckets are considered), firm can earn more profit. After 100 buckets, the profit gain is not significant. Hence we set $T = 100$ buckets in the rest experiments since larger T does not bring in much gain, but extra overheads in computation.

Varying α. In this section we experiment the effect of α on the firm's utility. When $\alpha = 0$, the firm's main objective is all about the profit. It would try to extract as much utility from consumers as possible. On the other side, $\alpha = 1$ represents an objective toward promoting the products as firm exchanges loss with consumers' satisfaction (measured by surplus). Besides the profit and surplus, we also report the deadweight loss and the combined

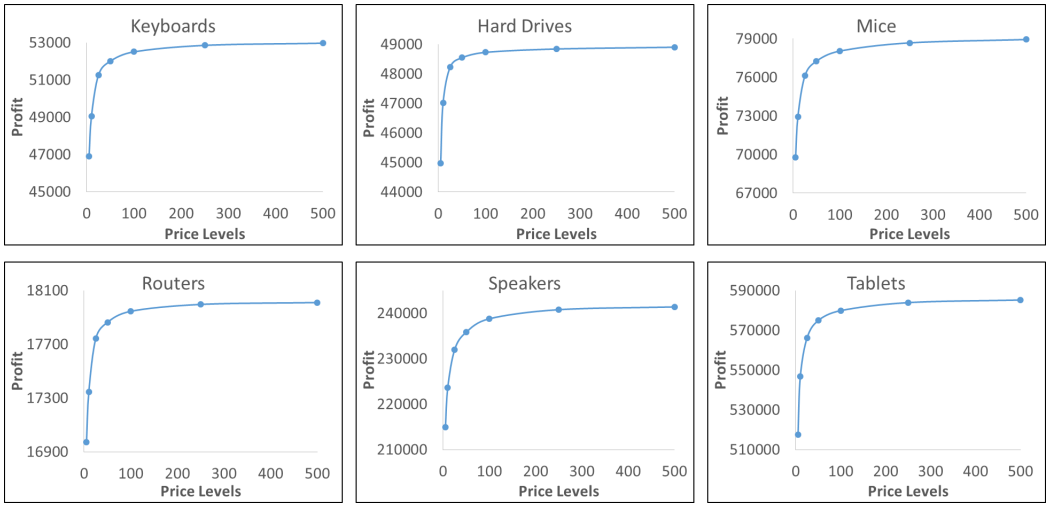


Figure 6.1: Effect of different number of price buckets on firm's profit.

utility (following Equation 6.1).

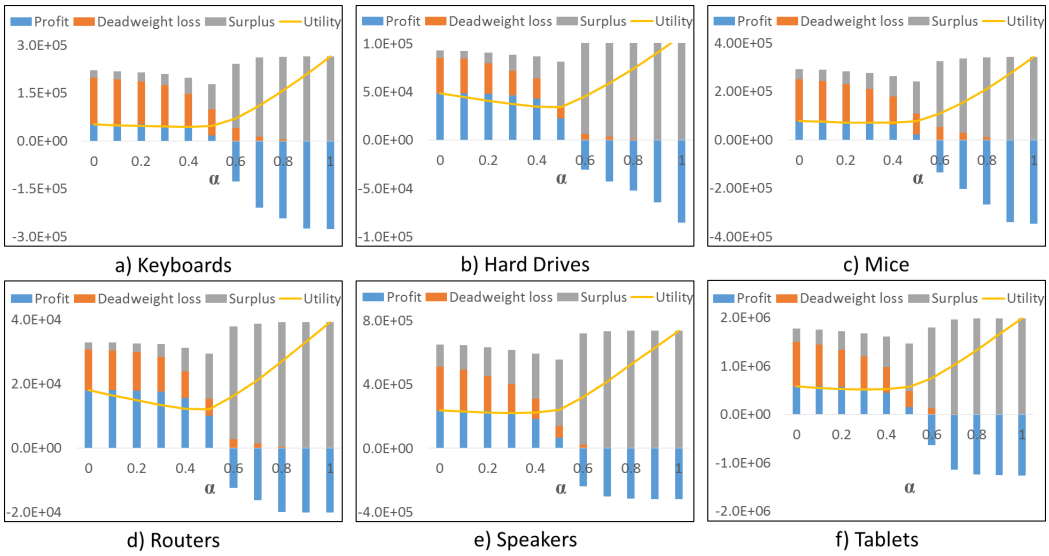


Figure 6.2: Effect of different α values on firm's utility.

Figure 6.2 shows the trends of four factors, i.e., firm's utility (labeled as *Objective*), profit, deadweight loss and surplus, on 6 sub-categories with different α. The horizontal axis represents different α values. The vertical axis reflects the firm's utility (in dollar unit). As α has a direct impact on determining the optimal price, all the four factors are affected differently with various α. The impact is consistent across sub-categories.

- **Profit.** At $\alpha = 0$, it is straightforward to see the peak of the profit.

When α increases, the profit declines and even becomes negative ($\alpha = 0.6$), signaling a profit loss to the firm. We reason that increasing α eventually leads to reduction in optimal price. It is when the price is lower than the product cost that firm starts to have loss.

- **Surplus.** The surplus is positively correlated with α. With large values of α, most of the product prices are zero. It explains the large amount of surplus the firm can obtain when $\alpha \geq 0.8$ For products priced at zero, the extracted surplus is the total willingness-to-pay.

- **Utility.** As a weighted combination between profit and surplus, the trend of the utility can be explained based on both quantities. The utility decreases with $0 \leq \alpha \leq 0.5$ for the more contribution of the profit decrease than the surplus gain. Similarly, the increase in utility when $\alpha \geq 0.5$ can be explained by the more contribution of the surplus gain over the profit decrease.

- **Deadweight loss.** Deadweight loss follows the similar trend with the profit. It also keeps decreasing when α increases. Although the firm's utility function does not include deadweight loss, it is still affected by a negative correlation with surplus.

- There is a drastic change when α increases from 0.5 to 0.6. The profit becomes negative. The surplus gain is significant comparing to lower α values. And the utility starts to increase. We can interpret the phenomenon that for firms who value consumer surplus more than profits, they have to sacrifice their own profits.

As we do not observe any additional information signaling firm's business strategy, we assume in our study the firm is profit-oriented. Hence, α is set to zero for the remaining experiments.

6.4.3 Profitability Results for Bundles

So far we only experiment on the *Unbundling* strategy. In this section we compare and analyze the profitability of selling bundles in a configuration on two different perspectives, namely bundle size and compatibility. After listing out comparable methods and evaluation metrics, we begin with the effect of various bundle size constraints on profitability. The compatibility between components in bundles are discussed next.

Comparable methods. With a combination between two bundling form (pure bundling and mixed bundling), and two heuristics algorithms (matching and greedy), there are in total four bundling configuration strategies, namely *Pure_Matching* , *Pure_Greedy* , *Mixed_Matching* , and *Mixed_Greedy*

Evaluation metrics. As we assume our firm is profit-oriented, the evaluation metrics are centered around profit. Specifically, we track and compare the returned profit from each strategy listed above on the basis of *Unbundling* 's profit. We therefore define *Gain over Components*, i.e., the relative profit gain over the *Unbundling* strategy and use it as the main metric to evaluate each bundling strategy.

6.4.3.1 Varying Size Constraints

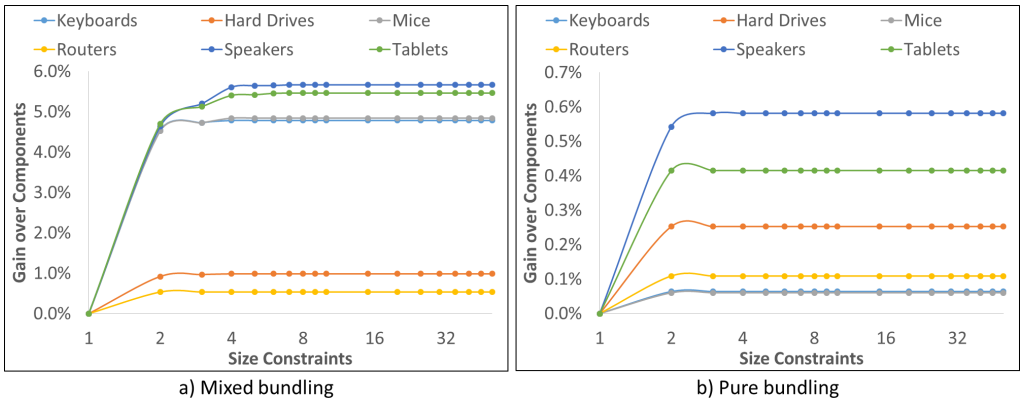


Figure 6.3: Effect of size constraints on firm's profit.

In theory, firm can choose to sell bundles of any size as long as it increases

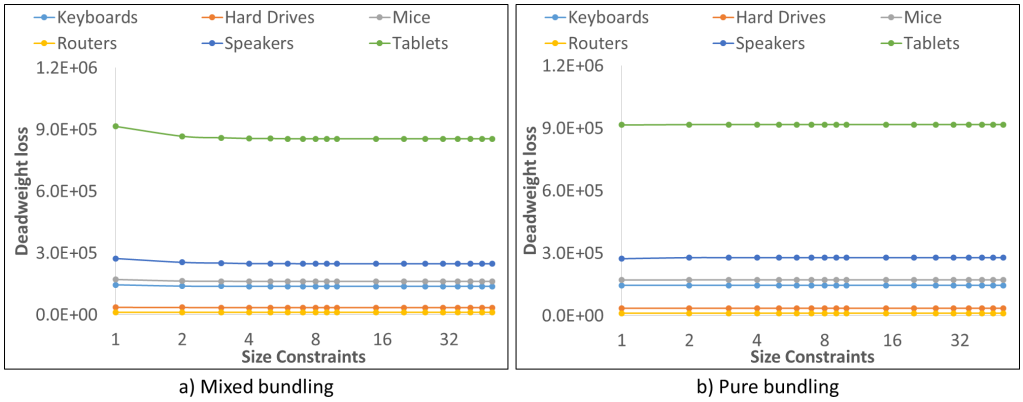


Figure 6.4: Effect of size constraints on consumer deadweight loss.

profit. In practice, firm may prefer packages at certain sizes due to production cost, competition, legality, etc. We now investigate the effect of constraining the maximum bundle sizes on each bundling form. We experiment various k's value. Figure 6.3 shows the results of matching-based algorithms with mixed and pure bundling. The greedy-based algorithms have similar observations.

Figure 6.3 contains two subgraphs displaying the trend of profit gain over *Unbundling* on mixed and pure bundling respectively. In each subgraph, the horizontal axis represents various size constraints. The vertical axis represents the relative gain over *Unbundling* on each product category. There are common observations for both bundling forms.

First, the profit increases with the size constraints, which applies on any bundling forms and product category. When $k = 1$, bundling is equivalent to the *Unbundling* strategy. For $k = 2$, bundling starts to gain over *Unbundling* . As we increase the size constraint, the profit keeps growing though at a slower rate. The decreasing trend of deadweight loss over bundle sizes in Figure 6.4 implies that more products are sold under bundles, resulting in higher profits. Previous work focuses mostly on bundles of size 2, and obtaining the optimal bundling configuration of size 3 and above is NP-hard. This experiment shows that while bundles of size 2 could produce some revenue gain, there is still a significant additional amount of revenue to be gained from bundles of larger sizes. This validates our approach in designing heuristic algorithms to discover

larger bundles.

Secondly, mixed bundling-based strategies returns higher profit than its pure bundling counterpart, which is well-established in literature. The difference varies across categories, e.g. 4.72% on Keyboards, 0.73% on Hard Drives, 4.78% on Mice, 0.43% on Routers, 5.09% on Speakers and 5.05% on Tablets.

Figure 6.5 shows the trend of profit gain over *Unbundling* for mixed bundling on rating-based willingness to pay. Compared to Figure 6.3a, the same observations can be drawn as above. In general, mixed bundling still have higher profit gain over *Unbundling* .

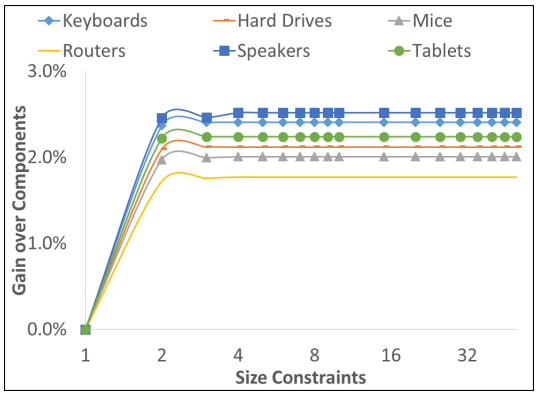


Figure 6.5: Effect of size constraints on firm's profit under mixed bundling (rating-based willingness to pay).

For pure bundling, we have to lower the variable cost from 70% of the Amazon listed price to 10% of the price in order to have positive gain over *Unbundling* , as shown in Figure 6.6. It is natural that high product cost would cut down on profit.

6.4.3.2 Bundling Compatibility

Although in this section we only show some examples of applying compatible rules on our proposed methods, they can also work well with other rules defined by terms in practice, as long as the rules can be represented as described in Section 6.3.1.2. For illustrative purposes, we merge all the consumers and products from the six sub-categories into Electronics category. We define two

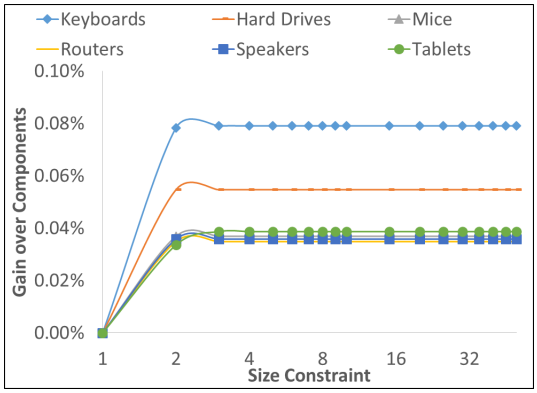


Figure 6.6: Effect of size constraints on firm's profit under pure bundling at lower cost(rating-based willingness to pay).

.

	Cross-category	Within category	Unrestricted
Mixed_Matching	1.45%	5.13%	5.49%
Mixed_Greedy	1.39%	4.74%	5.04%
Pure_Matching	0.28%	0.40%	0.65%
Pure_Greedy	0.28%	0.39%	0.66%

Table 6.4: *Gain over* Unbundling with/without compatible rules

rule sets $\mathcal{L}_1$ and $\mathcal{L}_2$ on the combined category as follows. The first rule set $\mathcal{L}_1$, called Cross-category, is to form bundles of products from different categories. As we have six sub-categories, the largest possible bundle is of size 6, consisting components from each category. The second rule set $\mathcal{L}_2$, called *Within category*, on contrary is to form bundles of products within the same category. Note that as the compatible rules constraint the number of possible candidates, we expect to see a decrease in profit gain in presence of such rules. Table 6.4 shows the *Gain over Components* of using $\mathcal{L}_1$ and $\mathcal{L}_2$ on the Electronics category. As expected, the Cross-category rule results in a small gain in profit as we cannot form a lot of bundles. Meanwhile, the *Within category* rule yields a profit gain close to *Unrestricted* (i.e., no rules). This is reasonable as we have more bundle candidates to consider in both cases.

6.4.4 Computational Efficiency Results for Bundles

Below, we discuss the efficiency of our matching-based and greedy algorithms. Timing is based on an Intel Core i7-4770 3.40GHz machine with 12GB RAM.

6.4.4.1 Optimal versus Heuristics

We now compare to the weighted set packing's optimal solution (*Optimal*) described in Section 6.3.4, as well as the greedy solution with known approximation bound (*Greedy_WSP*) in Section 6.3.5. Our objective is to show that our tractable algorithms can reach close to the optimal solution, and yet with greater efficiency. The following comparison includes only pure bundling as the reduction to weighted set packing is only defined for pure bundling.

Because weighted set packing requires enumeration of all subsets of items, it is only tractable for small problem sizes. To produce a small-scale dataset, we randomly select $N \in \{10, 15, 20, 25\}$ items from the merged Electronics data set, but include all the users. Even for this small-scale data set, the enumeration and revenue computation for $2^N - 1$ subsets of items require a significant amount of time as reflected in Table 6.5. Note that $2^N - 1$ is just the number of potential bundles, and finding a solution, i.e., a set of pair-wise non-overlapping bundles covering all items and having the maximum profit, requires another exponential search in this space. It is prohibitive for any larger number of items, and is not practical in real scenarios.

We create 20 such small-scale data sets in the following experiments. The average results are reported.

Comparison to Optimal. In Table 6.6, we compare *Pure_Matching* and *Pure_Greedy* to the weighted set packing solutions in terms of revenue coverage. Notably, for sample sizes of 10 to 20 our algorithms reach the same revenue coverage as *Optimal* for all the random samples. This could well be due to the relatively small sample sizes, but the *Optimal* solution can only be computed for small sample sizes.

	Running Time (seconds)			
	$N = 10$	$N = 15$	$N = 20$	$N = 25$
Electronics	< 1	< 1	63.85	2485.85

Table 6.5: Subset generation time

	Profit (dollar)			
	$N = 10$	$N = 15$	$N = 20$	$N = 25$
Pure_Matching	4.88E+03	1.02E+04	1.58E+04	1.68E+04
Pure_Greedy	4.88E+03	1.02E+04	1.58E+04	1.68E+04
Optimal	4.88E+03	1.02E+04	1.58E+04	-
Greedy_WSP	3.37E+03	7.62E+03	1.20E+04	1.26E+04

Table 6.6: Comparison to Weighted Set Packing on Electronics

Table 6.7 shows how our algorithms are much more efficient than *Optimal*. The running time of *Optimal* grows extremely fast. Importantly, this running time has not even included the time to enumerate all the subsets, which may reach to 1 hour for 25 items. Even so, the result for 25 items cannot be computed due to insufficient computing resources to process the $2^{25} - 1$ or 33 million boolean variables required by the ILP. Extending *Optimal* to even larger N is not feasible for the available computational resources.

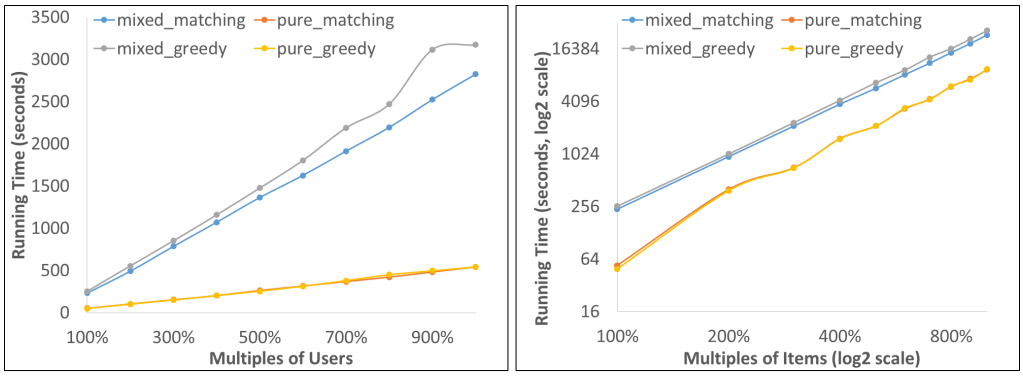
Comparison to Greedy_WSP. Table 6.6 also shows that *Pure_Matching* and *Pure_Greedy* outperform the current approximation solution for weighted set packing *Greedy_WSP* (with known approximation factor of $\sqrt{N}$). Our methods have higher profits across different sample sizes from 10 to 25. Empirically, it is evident that our algorithms reach a better approximation of *Optimal* than *Greedy_WSP*.

This is achieved at greater efficiency as well. Table 6.7 shows that not only *Greedy_WSP* takes more time, but the running time also increases significantly. For $N = 25$, our algorithms complete in less than a second. *Greedy_WSP* requires around two minutes on average (which would be much worse if we include the time for subset enumeration beforehand).

As our algorithms outperform the *Optimal* and *Greedy_WSP* in terms of speed, we exclude the last two in the next section, where we scale the data up

	Running Time (seconds)			
	$N = 10$	$N = 15$	$N = 20$	$N = 25$
Pure_Matching	0.044	0.070	0.093	0.111
Pure_Greedy	0.043	0.068	0.095	0.111
Optimal	0.007	0.802	42.530	-
Greedy_WSP	0.000	0.138	4.313	114.642

Table 6.7: Comparison to Weighted Set Packing on Electronics



a. Multiples of Users b. Multiples of Items

Figure 6.7: Scalability of Bundling Algorithms

to thousands of consumers and products.

6.4.4.2 Scalability

To investigate the scalability of the proposed algorithms, we measure how their running times are affected by the number of consumers, as well as by the number of products.

To create larger datasets with the same number of products, but varying number of consumers, we clone the consumers in the Computer Peripheral category using a multiplication factor. For example, the original dataset has a factor of 100%. For the factor of 200%, we have the same number of items, and twice as many users (with the same ratings as the original users). Figure 6.7(a) shows how running time varies with different multiplication factors. All the algorithms scale linearly with the number of users, which is reasonable since it only affects searching for the optimal price, which is $O(M)$ (see Section 6.2.2).

Figure 6.7(b) shows the scalability with respect to different multiples of products. We apply the same multiplication as we do on the consumer-side.

Both axes are in $\log_2$ scale. The linear growth in running time resembles a polynomial trend to the number of products, which is consistent with the complexity analysis in Section 6.3.3. The matching algorithms are more efficient than the greedy algorithms, because greedy requires more iterations to converge, since in each iteration it only creates one bundle.

6.5 Related Work

Early work on bundling focuses on explanatory models of profitability brought by selling bundled product. One of the most widely-accepted explanation is due to consumer's negatively correlated valuation on different products. [97] first gives a simple example on two buyers A and B and two movies X and Y, in which A prices X at higher price than B does, but would pay less on Y than B. [1] develops the example into an analytical model with two products, showing different consumer segmentations based on their reservation prices on the pair and pointing out which segments contribute to the extra profits by selling bundles. Besides, the authors introduce two different bundling strategies, namely pure bundling (i.e. selling only bundles) and mixed bundling (i.e. selling both bundles and the corresponding components). Based on the model, their main conclusion of bundle's profitability is for its ability to target and extract surplus from different consumer segmentations. Following [1, 90] represents consumer reservation prices as a bivariate normal distribution, and also re-confirms the profitability of bundling strategy under negative correlation in consumer valuation. As the main purpose is to reason profitability from bundling, these work leave several open research problems such as bundle design and pricing, inferring consumer valuations on bundles and components, etc. [106]. Due to its practicality, the problem of optimal bundle design and pricing based on consumer demand also has drawn a lot of attention from researchers.

Studies on bundle design and pricing deal with two fundamental research questions: 1) What are the products that can be bundled and under what bundling form (*pure* or *mixed*) ? and 2) How to price these bundles to maximize the total profit ? Relevant studies in literature revolve around these two questions from different perspectives. In the following, we summarize and contrast each direction with our work.

Connection to our study. One research direction is to examine bundle feasibility based on the relationship among components of a bundle. For example, [104] studies the bundle design under three types of relationships, namely independently valued products (e.g. a kitchen tool set[5]), complements (e.g. a package of guitar and instructional book with CD), and substitutes (e.g. a 6 pack of coke can).

Our work has different objectives than [104]. First, rather than explanatory purpose, our main goal is to tackle the computational challenge in bundle design. From a given product set, we seek for a collection of bundles at different sizes maximizing firm's objectives in total, instead of feasibility of 2-bundles (i.e. bundles of two components). Secondly, we allow relationship between components as part of the inputs so that the outputs consist of "feasible bundles" regarding to firm's desire.

In the scenarios where relationship among product is absent, [74] proposes a probabilistic model to infer complementary and substitutable products from their online reviews and metadata. Our work is also different from [74]. The latter work only focuses on relationship among products, rather than pricing and computational aspect in bundling. We treat this work to be orthogonal to our problem, since the information on product relationship can be integrated to produce compatible bundles (Section 6.4.3.2).

Another direction is to examine bundle feasibility on a large number of products. Given a set of products with zero or very low marginal cost, [9]

[5]https://www.oxo.com/products/cooking-baking/kitchen-tool-sets/everyday-kitchen-tool-set-15pc

proposes a probabilistic analytical model showing that selling a bundle of large number of information good for a single price can bring higher profits than selling the same goods separately. Consumer valuations on components are assumed to depend not on its nature, but rather on the size of bundle it belongs to. Although their study focuses on bundles of large size, the information goods are treated identically. This assumption drastically reduces the number of bundles in consideration from an exponential to a linear number in relative to the cardinality of the inventory. For N goods, there are at most N bundles which are different in size. Our work does not need this assumption. Another difference is that the study limits to information goods, meanwhile our work can work with physical goods of variant marginal costs.

Yet another direction is to examine bundle feasibility from the consumer response data [45]. Given a set of products (components and bundles), a set of pricing schemes on these products, and responses on these pricing schemes from consumers, the authors propose a probabilistic model capturing the latent reservation prices of each consumer on each product. The model can be used to estimate the reservation price of each consumer on each product, as well as compute the pricing scheme yielding highest profit in expectation. As the computational complexity is not of their focus, they only experiment in a very limited number of products and bundles.

The most similar work to ours is to examine bundle feasibility from pricing perspective [36]. Given a set of bundles (of size one or more), a set of consumer segments and their reservation prices on every bundle, the goal is to determine the price for each bundle so that firm can earn maximum profit when it offers all the given bundles at the computed pricing scheme. Because each consumer is assumed to pick the bundle maximizing their surplus, the work has to consider all the possible candidates, which is up to $2^N - 1$ bundles from N distinct components. The authors formulate the problem as a Mixed Integer Programming (MIP) with maximizing firms profit as the objective function.

When N is large, this approach faces two problems, namely storage problem (i.e. create $2^N - 1$ decision variables, one for each bundle) and computational problem (i.e. consider $2^N - 1$ bundles for each customer segment to determine the surplus-maximizing one). In an effort to reduce the number of decision variables, the authors provides a relaxed problem from the original MIP with a significantly small number of variables. However, their relaxed version still suffers from the computational issues as it has to implicitly consider all possible bundle candidates.

Our work can be distinguished from [36] in two perspectives, namely the problem objective and solution. First, at the problem level, the latter focuses only on pricing all the possible bundles, meanwhile we have two targets: 1) Select the most profitable *bundling configuration* (i.e. a collection of bundles) out of all the possible *bundling configuration*s and 2) Price each bundle in the selected *bundling configuration*. Second, from the computational perspective, the approach in [36] is exhaustive search, leading to the optimal solution. As the exhaustive approach is intractable on large scale data, we approach the problem with heuristic solutions. With extensive experiments in Section 6.4.4 to compare the results from ours against the optimal method, we show that the discrepancies between the two are negligible on small datasets. Moreover, our methods can scale up to large-scale data sets with tens of thousands of products and hundreds of thousands of consumers.

6.6 Summary

We address the bundle configuration problem by mining willingnes- to-pay from user-generated online reviews. This is a major difference from the traditional business approach where explicit solicitation from consumers is frequently required, which does not work in large scale. Our objective is to determine the bundle configuration that maximizes the total revenue. We address two

variants of this problem, based on the respective bundling strategies of pure bundling and mixed bundling. We show that the problem is NP-hard for pure bundling of size 3 or more, and introduce a couple of effective heuristic solutions based on graph matching and greedy selection respectively. Experimentation shows that our approach results in higher revenues than individual components as well as bundles based on frequent itemsets. Moreover, the matching-based algorithm outperforms the greedy algorithm in effectiveness and efficiency.

Chapter 7

Finding Profit Gain-Maximizing Top-K Diverse Bundles

7.1 Overview

In Chapter 6, we focus on the problem of a seller who is interested in finding a profit-maximizing partition of the product set. Besides non-overlapping bundles, it is also common in practice that the seller offers overlapping bundles to the market, for example, fruit baskets. Moreover, the non-overlapping constraint may not always bring the most profitable bundling solutions to the seller. In this chapter we relax that constraint. Since not all the bundles are well-received by consumers, some bundles return less profit than selling their components. It is natural for the seller to seek the ones with highest profit gain. Without loss of generality, we assume the seller is interested in the top-K profit gain bundles, $K \geq 1$.

Example. Let us reconsider the example of three products A, B and C in Example 6.3. There are four possible bundles, namely $(A, B), (A, C), (B, C)$ and (A, B, C). The profit gain of (A, B) over selling its components is $\$30 - \$16 - \$11 = \$3$. Similarly, the profit gain of (A, C), (B, C) and (A, B, C) is $\$5$, $-\$2$, and $\$12$, respectively. If the seller is interested in top-2 bundles with

the most profit gain, the solution should be (A, B, C) and (A, C).

7.1.1 Problem Statement

We first discuss the the main aspects of the problem, before introducing its formal definition. To evaluate profit of a given bundle b, we consider the same problem settings as described in Section 6.1.1. Given consumer's willingness-to-pay data W, variable cost for every product, assumptions on consumer's purchase behaviors, we use $f(b)$ (Equation 6.8 with $\alpha = 0$) to indicate profit of selling b.

Profit Gain. By $gain(b)$ we denote a profit gain function which estimates how much more profitable selling b is compared to selling its components. Recall from Chapter 6 that there are two bundling forms, namely pure bundling and mixed bundling. For a given bundle b, each bundling form would result in a different profit. In this chapter we consider pure bundling. Mixed bundling is of future direction.

Under pure bundling, a seller can offer b or its components to the market. The profit gain of b is measured on the difference between the profit from selling b and the one from selling its components. We use the two following instances of $gain(.)$

$$\begin{aligned} \text{Absolute gain: } gain(b) &= \mathcal{P}(b) - \sum_{i \in b} \mathcal{P}(i) \\ \text{Relative gain: } gain(b) &= \frac{\mathcal{P}(b)}{\sum_{i \in b} \mathcal{P}(i)} - 1 \end{aligned} \tag{7.1}$$

We set $gain(b) = 0$, when b is an empty set or contains only one product. When $gain(b) > 0$, b is said to be more profitable than its components. Otherwise, b is less profitable than its components. For the rest of this chapter, by $gain(.)$ we mention the absolute gain in general context, unless otherwise specified.

Lemma 1. *The profit gain function $gain(.)$ is not sub-modular and non-monotone.*

PROOF. We prove by a counter example. By definition, if $gain(.)$ is a submodular function, for every $b_1, b_2 \subseteq I$, we have $gain(b_1) + gain(b_2) \geq gain(b_1 \cup b_2) + gain(b_1 \cap b_2)$ [58]. In Table 6.3, $gain(\{A\}) = gain(\{B\}) = 0$. However, $gain(\{A, B\}) = \$3 > 0$, which violates the definition of submodular functions. For monotonicity, although B both belong to two bundles (A, B) and (B, C), $gain(\{A, B\}) = \$3 > 0$ but $gain(\{B, C\}) = -\$3 > 0$.

Likewise, we can also prove $gain()$ with the relative gain is not submodular and non-monotone. ■

LEMMA 1 implies that a bundle at large size does not necessarily have more profit gain than any of its subsets. Hence, we have to consider the search space of all bundles at all sizes.

Top-K Profit Gain Bundles. By $\mathcal{S}$ we denote a set of all possible bundles, D' as a subset of K bundles, i.e., $D' \subseteq \mathcal{S}, |D'| = K$. Given the $gain()$ function, seeking the top-K profit gain bundles forms the following optimization problem

$$D = \underset{D' \subseteq \mathcal{S}, |D'|=K}{\operatorname{argmax}} \sum_{b \in D'} gain(b) \tag{7.2}$$

We call this problem the *top-K profit gain bundles* problem, or in short the top-K bundles problem.

Diversity. We allow D to contain overlapping bundles. Since it is possible for two highly overlapping bundles to have high profit gain, for example, two bundles (A, B, C) and (A, C) in the above example, we let the seller to decide whether they want such bundle pairs in D. The diversity of a bundle set is therefore determined from the extent of similarity between bundles in the top-K solution as follows.

We consider a common setting of diversity aware search in the literature [7, 81]. A bundle set D is said to be diverse if its elements are similar to some certain degree. The similarity between any two bundles b_1 and b_2 is determined

by a similarity function $sim(b_1, b_2)$ and a similarity threshold θ which are user-specified. In general, $sim(b_1, b_2) \in [0, 1]$ and $\theta > 0$. The more $sim(b_1, b_2)$ is closer to 1, the more similar b_1 is to b_2. It is straightforward that $sim(b_1, b_1) = 1$. We have the following definitions.

Definition 1. BUNDLE SIMILARITY

Two bundles b_1 and b_2 are similar iff $sim(b_1, b_2) \geq \theta$. We denote $b_1 \simeq b_2$ when $sim(b_1, b_2) \geq \theta$. Otherwise, we denote $b_1 \neq b_2$.

Given Definition 1 on similarity between a bundle pair, we define a diverse set of bundles as follows [81].

Definition 2. DIVERSE BUNDLE SET

A bundle set $D \subseteq \mathcal{S}$ is said to be diverse on a given similarity function $sim() \in [0, 1]$, and a similarity threshold $\theta > 0$ iff $\forall b_1, b_2 \in D, b_1 \neq b_2$.

Without loss of generality, a set of a single bundle is always diverse. Now we proceed to formulate the profit gain maximizing top-K diverse bundles problem.

Notation. We reuse some of notations in Section 6.1.1 such as W to indicate the willingness-to-pay matrix, $\boldsymbol{c}$ to indicate a cost vector for every product, we introduce additional notations particularly related to the top-K problem.

- L is a positive integer to indicate the maximum components a bundle can have.
- A bundle set $\mathcal{S}_L$ contains all possible bundles having at most L components constructed from a given product set.
- $sim()$ is a user defined similarity function, $sim() \in [0, 1]$
- A similarity threshold $\theta > 0$
- K is a positive integer to indicate the maximum size of D.

Problem 3. PROFIT GAIN MAXIMIZING TOP-K DIVERSE BUNDLES

*Given W, $\boldsymbol{c}$, L, K, $sim()$, and θ, we seek a bundle set $D \subseteq \mathcal{S}_L$ such that D is diverse (*DEFINITION *2) and satisfy, i.e.,*

$$D = \underset{D' \subseteq \mathcal{S}_L, |D'| \leq K}{\operatorname{argmax}} \sum_{b \in D'} gain(b) \tag{7.3}$$

Due to LEMMA 1, we have to consider a search space $\mathcal{S}_L$ of all the bundles having up to L components. D would have less than K bundles if there are less than K diverse bundles with positive profit gain in $\mathcal{S}_L$. When D has K bundles, by Equation 7.3, D is also a top-K bundles with highest profit gain.

Example. In Example 6.3, it depends on how the seller specifies $sim()$ and θ, the top-2 profit gain bundles can be different. Assume $sim()$ is the Jaccard distance. When $\theta = 0.01$ (high diversity), the solution is (A, B, C). When $\theta = 0.7$ (high similarity), the solution consists of (A, B, C) and (A, C).

7.2 Complexity Analysis

The PROBLEM 3 is NP-Hard, even with $K = 1$. To prove its hardness, let us first consider the **decision** version of the DIVERSE TOP-1 ABSOLUTE GAIN problem, which is defined in the PROBLEM 4. For simplicity, we reuse the same notations in PROBLEM 3.

Problem 4. DIVERSE TOP-1 PROFIT GAIN *(Decision)*

Given W, $\boldsymbol{c}$, L, a positive value δ, is there any bundle $b \subseteq \mathcal{S}_L, b \neq \emptyset$ such that $gain(b) \geq \delta$.

Lemma 2. PROBLEM *4 is NP-Complete.*

PROOF. We show that the NP-Complete problem Vertex Cover is a special instance to the PROBLEM 4. Given a graph $G(V, E)$ with vertex set V and edge set E, and a positive integer L', the Vertex Cover problem is to confirm

the existence of a subset $S \subseteq V, |S| \leq L'$ which contains at least one end point of every edge $e \in E$.

To transform an instance of the Vertex Cover problem to PROBLEM 4, we introduce two quantities α and β satisfying

1. $\alpha, \beta > \epsilon > 0$
2. $1 < \frac{\alpha-\epsilon}{\beta-\epsilon} < \frac{M+1}{M} = 1 + \frac{1}{M} < 1 + \frac{1}{M-1} = \frac{M}{M-1}$

In addition, we introduce a dummy vertex v_d with dummy self-loop edge $e_d = (v_d, v_d)$ into G to create $G' = (V', E')$, with $V' = V \cup v_d, E' = E \cup e_d$.

The transformation from an instance of Vertex Cover to the PROBLEM 4 is as follows.

- $V' \rightarrow I$, i.e., each vertex is treated as a product. We have $N = |V| + 1$ products in total, including the dummy node v_d. We assume the variable cost for every product is ϵ.
- $E' \rightarrow U$, i.e., each edge is treated as a consumer. We have $M = |E| + 1$ consumers in total, including the self-loop dummy edge e_d. Since each edge involves with two nodes, each consumer is only interested in buying two products with a willingness to pay of α. In other words, $\forall e = (i, j) \in E, w_{e,i} = w_{e,j} = \alpha$. For the self-loop edge, $w_{e_d,v_d} = \alpha$.
- The remaining entries of W are set to β.
- We set L equal to $L' + 1$.
- δ is set to zero. Basically we seek bundles with positive absolute profit gain.

The following is derived from Equation 6.8 with $\alpha = 0$.

- Profit of every vertex: $M(\beta - \epsilon)$. There are two candidate prices, namely β or α. Since each vertex, including v_d, incidents to at most $|E| = M - 1$

edges, and $M(\beta-\epsilon) > (M-1)(\alpha-\epsilon)$ by definition, the optimal price is therefore β.

- Profit of selling components of a bundle b of l vertices (or Components): $lM(\beta-\epsilon)$.
- For a set of vertices, an edge may have zero, one or both ends in the selected set. Hence, the total willingness to pay that each edge contributes to b is one of the following
  - $\beta + \ldots + \beta + \beta = l\beta$
  - $\beta + \ldots + \beta + \alpha = (l-1)\beta + \alpha$
  - $\beta + \ldots + \alpha + \alpha = (l-2)\beta + 2\alpha,\ \forall l \geq 2$
- There are three price candidates for b, namely $p_1 = l\beta, p_2 = (l-1)\beta + \alpha$ and $p_3 = (l-2)\beta + 2\alpha$. Since $\beta < \alpha$ by definition, $l\beta < (l-1)\beta + \alpha < (l-2)\beta + 2\alpha$. Hence, $p_1 < p_2 < p_3$.
- The number of consumers would purchase b under each price in the price list: M, M_2 and M_3. Since $p_1 < p_2 < p_3$, $M \geq M_2 \geq M_3$.

Lemma 3. *Any price other than $p_1 = l\beta$, which there are less than M consumers who can afford, would lead to less profit.*

PROOF. Assume that there are $M-1$ consumers who can afford the largest price option, p_3. The equivalent profit is $(M-1)((l-2)\beta + 2\alpha - l\epsilon)$. Since $(M-1)(l-2)\beta < M(l-2)\beta$, $(M-1)\alpha < M\beta$ (by definition), we can see that $(M-1)(p_3-\epsilon) < lM(\beta-\epsilon)$. ■

A solution to the transformed instance of Problem 4 is a bundle of at most $L'+1$ vertices whose profit is larger than $lM(\beta-\epsilon)$. With Lemma 3, the optimal price has to be larger than $l\beta$ and all the edges have at least one end in the selected bundle. To cover the self-loop edge e_d, the returned bundle has to include the dummy node v_d. Other than that, the remaining nodes have to

cover all non- self-loop edges. By removing v_d and e_d, we arrive at a solution to the original Vertex Cover instance. If no solution with positive gain can be found, the Vertex Cover also has no solution. ■

Lemma 4. *The* PROBLEM *3 is NP-Hard.*

PROOF. When $K = 1$, we can prove the PROBLEM 3 is NP-Hard by using LEMMA 2. Hence, for all $K > 1$, the problem is NP-Hard. ■

7.3 Methodology

To construct D from a given product set, a principled approach is to evaluate the profitability of all possible L-sized bundles using $gain(.)$, and select among all possible diverse bundle sets the one with maximum profit gain. We formulate this approach into a two-phase solution. In the first phase, a bundle set is generated. From this bundle set, we select the top-K bundles in the second phase. Recall that $\mathcal{S}_L$ (in PROBLEM 3) is a set of all possible bundles of at most L components. By $\mathcal{S}$ we denote a subset of $\mathcal{S}_L$. With the objective function in Equation 7.3, each phase of the solution has the following objective function.

Phase 1: Candidate Generation

$$\max_{\mathcal{S} \subseteq \mathcal{S}_L} \mathcal{F}(\mathcal{S}) \tag{7.4}$$

Phase 2: Candidate Selection

$$\mathcal{F}(\mathcal{S}) = \max_{D' \subset \mathcal{S}, |D'| \leq K, D' \text{ is diverse}} \sum_{b \in D'} gain(b) \tag{7.5}$$

The overall solution is depicted in Figure 7.1. In the following we discuss the approach to each phase of the solution.

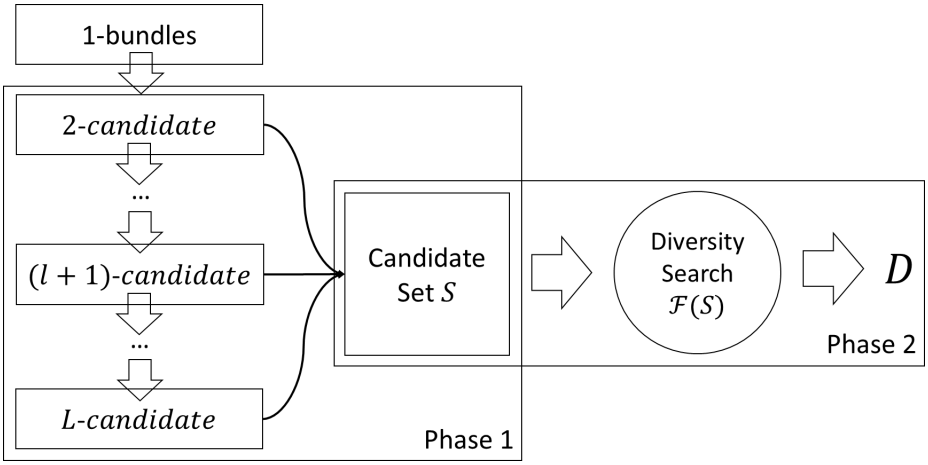


Figure 7.1: Overview of the two-phase solution

7.3.1 Phase 1: Candidate Generation

The problem in Phase 1 is to generate $\mathcal{S}$ from the given set of N products I. Generating $\mathcal{S}$ is challenging for two observations. First, an exhaustive enumeration of all possible bundles to construct $\mathcal{S}$, i.e., $\mathcal{S} = \mathcal{S}_L$, poses high memory and computational cost, especially when both N and L are large. Secondly, $\mathcal{S}$ has to include bundles at all sizes (up to L) due to the non-monotonicity of *gain*() (LEMMA 1). A desired $\mathcal{S}$ should include both diverse and profitable bundles, and at the same time should not be too large.

Here, we adapt an iterative approach commonly used in itemset mining algorithms to construct $\mathcal{S}$. Specifically, we iteratively construct size-l bundles from size-$(l-1)$ bundles by adding every product to each of the latter ones, until size-L bundles are generated. To reduce the computational and memory cost, we retain only a subset of size-l bundles at every iteration, and use the selected ones in the next iteration. By *l-candidate* we denote the set of selected size-l bundles. The final candidate set $\mathcal{S}$ is a union of all the *l-candidate*, $l = 2 \ldots L$.

We seek a pruning strategy to construct *l-candidate* at every iteration. A direct approach is to retain top size-l bundles with highest profit gain[1]. However, focusing on top profit gain bundles does not guarantee the diversity in the final $\mathcal{S}$. Hence, both diversity and profitability should be considered together

[1]This can be achieved by using a min-heap data structure.

in the pruning strategy. Denote *l-buffer* as a set of all generated distinct size-l bundles, B as a subset of *l-buffer*, $score()$ as a function to evaluate both the diversity and profitability of a bundle set, $K_{\text{candidate}}$ as a user-defined size of *l-candidate*. Ideally, *l-candidate* should be selected as follows

$$l\text{-}candidate = \underset{B \subseteq l\text{-}buffer, |B|=K_{\text{candidate}}}{\operatorname{argmax}} score(B) \tag{7.6}$$

To evaluate $score(B)$, all bundles in B are first sorted in descending order of their profitability measured by $gain(.)$. To prioritize bundles with large profit gain in B, we modify the Discounted Cumulative Gain (DCG) measure in the information retrieval

$$score(B) = \sum_{k=1}^{K=|B|} \frac{gain(b_k)}{log_2(k+1)} \tau(k, B), \tag{7.7}$$

where b_k is the k-th bundle in S, $\tau(k, B)$ is a cost factor for similar bundles. Basically, the more profitable bundles are in B, the larger $score(B)$ is. Moreover, to integrate the diversity, we discount the profitability of the k-th bundle in B if it is similar to the its $k-1$ predecessors in B. We set $\tau(1, B) = 1$. For $k > 1$, we explore three different types of $\tau(k, B)$.

- Selection by Top (TopSel) or no discount on similar bundles

$$\tau(k, B) = 1 \tag{7.8}$$

- Selection by Max (MaxSel)

$$\tau(k, B) = \max_{i<k}(1 - sim(b_i, b_k)) \tag{7.9}$$

- Selection by Arithmetic mean (AriSel)

$$\tau(k, B) = \sum_{i=1}^{k-1}(1 - sim(b_i, b_k)) \tag{7.10}$$

Algorithm 3 Candidate Set Generation

Input: $W, \boldsymbol{c}, L, K, K_{candidate}, sim()$
Output: Candidate Set S
Initialize 1-*candidate* to be a set of size-1 bundles.
Initialize $S = \{\}$
for $l = 2 \rightarrow L$ **do**
  Initialize l-*buffer*= $min\_heap(K_{candidate})$
  for $b \in (l-1)$-*candidate* **do**
    for $i \in I$ **do**
      Form a candidate size-l bundle $b' = b \cup i$
      Compute the profitability of b':
        $\Delta = gain(b')$ (Equation 7.1).
      if $\Delta \geq 0$ **then**
        l-*buffer* $\leftarrow$ l-*buffer*$\cup(b', \Delta)$
      end if
    end for
  end for
  Initialize l-*candidate* $= \{\}$
  while l-buffer$\neq \emptyset$ **do**
    $b' \leftarrow \text{argmax}_{b \in l\text{-}buffer}\ score(l\text{-}candidate \cup b)$
    l-*candidate* $\leftarrow$ l-*candidate* $\cup\ b'$
    l-*buffer* $\leftarrow$ l-*buffer* $\setminus\ b'$
    if $|l\text{-}candidate| = K_{candidate}$ **then**
      break.
    end if
  end while
  $\mathcal{S} \leftarrow \mathcal{S} \cup l$-*candidate*
end for
Return $\mathcal{S}$.

Depending on $\tau(k, B)$, we have different search strategy for l-*candidate*. When $\tau(k, B)$ is set to Equation 7.8 (i.e., no cost on selecting similar bundles), the l-*candidate* is simply the top-$K_{\text{candidate}}$ size-l bundles with highest $gain(.)$ values. With the other two cost factors, an optimal solution is to enumerate all possible subsets of l-*buffer* to select l-*candidate*, which can be costly. Alternatively, we first set l-*candidate* to be empty. Subsequently, we add one bundle from l-*buffer* to l-*candidate* such that the new l-*candidate* has maximum $score()$. The selected bundle is removed from l-*buffer*. The process is repeated until $K_{\text{candidate}}$ is added to l-*candidate*, or l-*buffer* is empty.

We can speed up the process further by pruning l-*buffer*. Instead of retaining all the generated size-l bundles, which is up to $O(NK_{candidate})$, we can remove the least profitable ones. Through empirical study, we find that re-

taining the list of $2K_{candidate}$ most profitable size-l bundles in *l-buffer* would be enough. Any larger number do not improve the result of Phase 2. The iterative process is summarized in ALGORITHM 3.

Here we analyze the complexity of the proposed solution in Phase 1. To retain the bundles with highest profit gain, we implement *l-buffer* as a min heap of $2K_{candidate}$ elements. The number of size-l bundles is N^2 when $l = 2$, and $2K_{candidate}N$ when $l > 2$. Since we add every generated size-l bundle to the heap, the complexity of adding size-l bundles to heaps is $O((N + 2(L - 2)K_{candidate})Nlog(2K_{candidate}))$. In addition, it takes $O(K^2_{candidate})$ to select *l-candidate* from *l-buffer*. Assume that $K_{candidate} = \psi K$, the overall complexity would be $O((N + 2\psi(L - 2)K)Nlog(2\psi K)) + O(K^2)$. Usually $L \times K \lll N$, we can simplify the complexity further to $O(N^2 log(2\psi K)) + O(K^2)$.

7.3.2 Phase 2: Candidate Selection

Given the candidate set $\mathcal{S}$ from Phase 1, the objective of Phase 2 is to select a subset D of top-K profitable and diverse bundles (Equation 7.5). Given a user-defined similarity function $sim()$ and a similarity threshold θ, searching for D is a specific instance of a framework work proposed in [81]. The problem is proved to be NP-Hard by a reduction from the maximum independent set. There are optimal solutions with efficient pruning strategies [81], as well as approximation algorithms with guaranteed bounds [50]. In this paper, we consider the optimal solution in [81] to search for D in Phase 2.

We transform the candidate set $\mathcal{S}$ to the input of the optimal algorithm as follows. Denote $G(V, E)$ is the input graph. Each node $v \in V$ corresponds to a bundle in $\mathcal{S}$. The weight of v is set equal to profit gain of the bundle, measured by $gain()$. There is an edge between two nodes u and v if the corresponding bundles are similar by $sim()$ and θ. The algorithm returns a set of at most K vertices with highest total weight, with no edges in the induced subgraph of the set. The corresponding bundles of the selected vertices are hence the

solution to the Equation 7.5.

7.4 Experiments

In this section, we first examine the performance of the proposed solution under various settings, and compare it with other existing methods. Moreover, we conduct several case studies in Section 7.4.5 to have better understanding on the top results.

7.4.1 Experimental Setup

Data. One of the worldwide e-Commerce websites is Amazon, which attracts a large volume of ratings and reviews from consumers. Amazon review data has also been used in research related to consumer preference [27, 74]. For our purpose of studying consumer preference at large scale, we consider 5-star rating data of ten Electronics sub-categories on Amazon [74]. We further process the data towards the bundling objective. We first trim off products which do not share any common raters with others. These products will not participate in any bundles, hence the removal will not affect to the final solutions. Next, we remove consumers with single rating. These consumers are interested only in one product, hence the bundling strategy is not effective on those consumers. Statistics of the preprocessed data is listed in Table 7.1. To increase the data scale, we combine all the sub-category data into one of $31,865$ consumers, $9,191$ products and $83,769$ ratings. The combined data is used through all the experiments in this section.

Similar to Section 6.4, here we also experiment with two types of willingness-to-pay data. The first type is derived from ratings as proposed in Section 5.1.1. The second type of willingness-to-pay data is derived from Chapter 5. We reuse the merged Electronics data in Section 6.4.3.2. Recall that the data has $20,336$ consumers, $5,042$ products and $45,023$ positive willingness-to-pay

Table 7.1: Data Statistics

Category	Number of consumers	Number of products	Number of ratings
Computer Speakers	2,734	604	6,159
External Hard Drives	4,719	653	11,727
Internal Hard Drives	2,167	564	4,964
Keyboards	3,954	1,262	8,851
Memory	2,912	1,250	6,538
Mice	5,604	1,213	12,590
Monitors	1,454	790	3,132
Routers	3,894	603	8,844
Tablets	4,442	1,057	10,052
USB	4,753	1,195	10,911
Combined data	31,865	9319	83,769

entries. Since similar observations can be drawn from both types of willingness to pay data, as shown in Figure 7.10 and Figure 7.11, we focus on the rating-based one in this section for its larger scale.

Metrics. Our end goal is to find the top-K diverse bundles with maximum sum of profit gain. For a bundle set D returning by a method, we evaluate its profitability by the following quantity.

$$sum\_of\_gain = \sum_{b \in D} gain(b) \tag{7.11}$$

It is straightforward that the higher sum_profit_gain is, the more profitable the solution can give.

Comparable methods. We compare the profitability of different candidate sets $\mathcal{S}$ returned from various approaches. One set of methods is the two-phase solution proposed in this paper, including different selection strategies $TopSel$, $MaxSel$ and $AriSel$. The other set of methods are baselines, including the closest work to our, the *BundlingConfiguration* algorithm in [27], and the two state-of-the-art itemset mining algorithms, namely frequent itemsets [23] and high utility itemset [100]. Under $BundlingConfiguration$, $\mathcal{S}$ forms a partition of the given product set. Under the itemset-based approaches, $\mathcal{S}$ contains frequent itemsets or high utility itemsets.

Problem Configurations. We experiment with different configurations related to the problem as follows.

- Production cost of a product is not often publicly available due to competitions among manufacturers. To cover the cost, the product price is usually set higher than its cost. In our paper we rely on the products' listed price on Amazon to estimate the cost. We set the cost to be a portion of the listed price, which is determined by a cost factor $\gamma > 0$. Various γ values are considered to capture different types of products, for example, low γ values indicate low-cost products. For simplicity, we apply the same γ to estimate the cost for every product.

- Since bundle is a set of products, we use Jaccard distance to measure the similarity between two bundles b_1 and b_2. In other words,

$$sim(b_1, b_2) = \frac{|b_1 \cap b_2|}{|b_1 \cup b_2|} \tag{7.12}$$

- We vary the similarity threshold $\theta \in \{0.2, 0.4, 0.6\}$.

- For size constraint on bundles, we vary L in $[2, 10]$.

- For the number of returned bundles, we vary K in $\{5, 10, 15, 20\}$.

By default, we set $\theta = 0.4$, $L = 10$, $K = 10$ and $\gamma = 0.1$. We vary each setting at a time, while fixing the others at their default values.

7.4.2 Comparison among Generation Strategies

In Phase 1 we propose three generation strategies for the candidate set $\mathcal{S}$ (Section 7.3.1). Here we evaluate how much more gain each strategy can bring under various sizes of each l-candidate set and the selection strategies ($TopSel$, $MaxSel$ or $AriSel$).

Varying number of candidate bundles. We examine how the size of the *l-candidate* sets $l > 1$ affects the total gain of the final solution. To keep it

simple, we assume all *l-candidate* share equal size. We set the size to be a factor of K. For example, with $K = 10$, we vary the size with five different values $\{10, 20, 30, 40, 50\}$. In Figure 7.2, we plot the correlation between running time (x-axis) and sum_of_gain (y-axis) across different candidate sizes for every selection strategies. The running time is recorded as the sum of generation time and selection time.

In general, the more candidates are generated, the more profitable solution we can find. At the same time, the corresponding running time also increases. When the candidate set is set to be four times of K, e.g., 40 when $K = 10$, the sum_of_gain already converges in most of the sub-figures in Figure 7.2, except the *AriSel* with absolute gain. However, with the size of 50, the trade-off between time and profitability becomes significant, in which more time is needed to obtain a negligible increase of sum_of_gain in return. This applies on both types of profit gain, as shown in Figure 7.2a and Figure 7.2b. Hence, we set each *l-candidate*'s set size to be four times of K in the remainder of the paper.

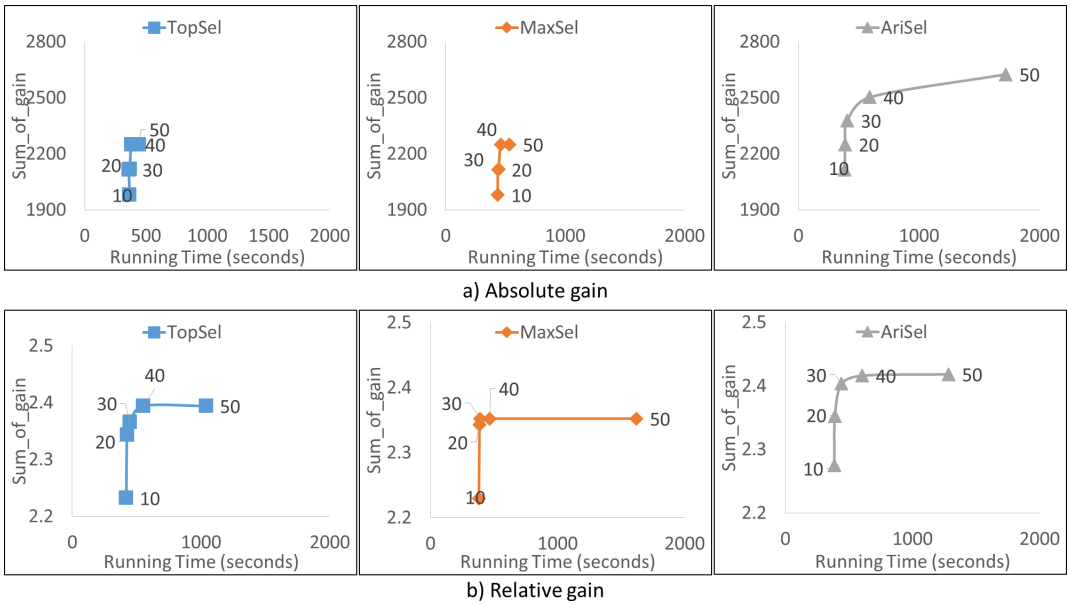


Figure 7.2: Effect of various candidate size ($\{10, 20, 30, 40, 50\}$) with $K = 10, L = 10$ on profit gain.

Among the selection strategies, *AriSel* returns the most profit gain top-K

result for every λ. When $\lambda = 4$, the solution of $AriSel$ gains more 11.26% than both $MaxSel$ and $TopSel$ (under $abs(.)$). With $rel(.)$, the gain is 0.92% larger than $TopSel$, and 2.71% larger than $MaxSel$. We reason the gain is due to the difference in the selection strategy of candidates. By degrading similar bundles during the selection process (Equation 7.10), $AriSel$ can explore more diverse bundles than $TopSel$. Comparing with $MaxSel$, the arithmetic mean-based selection is more effective in selecting both diverse and profitable bundles.

Varying selection strategies. As shown in Figure 7.2, the selection strategies also affects the top-K result. We investigate this effect on sum_of_gain across different L in Figure 7.3. Other settings are set to their default values. In terms of the absolute gain, $AriSel$ brings more gain than $TopSel$ and $MaxSel$ when $L > 5$ (Figure 7.3a). In terms of the relative gain, $AriSel$ only brings a marginal gain over $TopSel$ (Figure 7.3b). In general, each strategy returns more profitable results when L increases. This is as expected since the candidates at size L contains the ones at size $L - 1$.

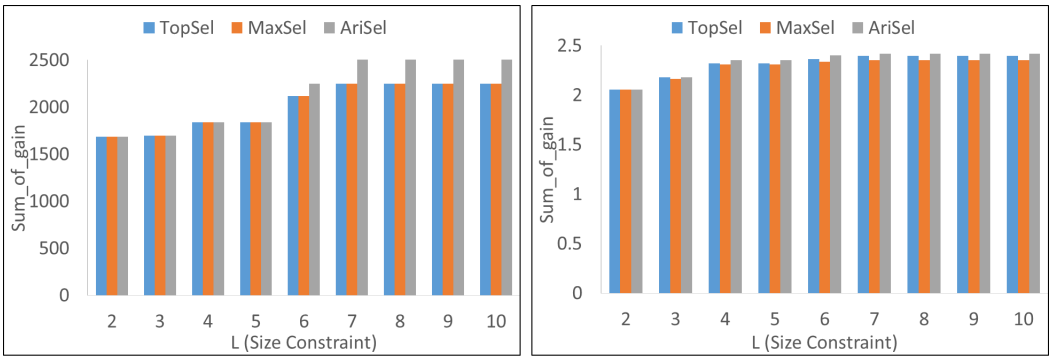


Figure 7.3: Effect of different candidate generation strategies on profit gain

We reason the difference among sum_of_gain comes from the diversity and profit gain of candidates in $\mathcal{S}$. To investigate this difference further, for each method we plot the average of profit gain of all bundles in its $\mathcal{S}$ at different size constraints (Figure 7.4), as well as the average of Jaccard distances of all bundle pairs (Figure 7.5).

In terms of absolute profitability, Figure 7.4a shows that the average of

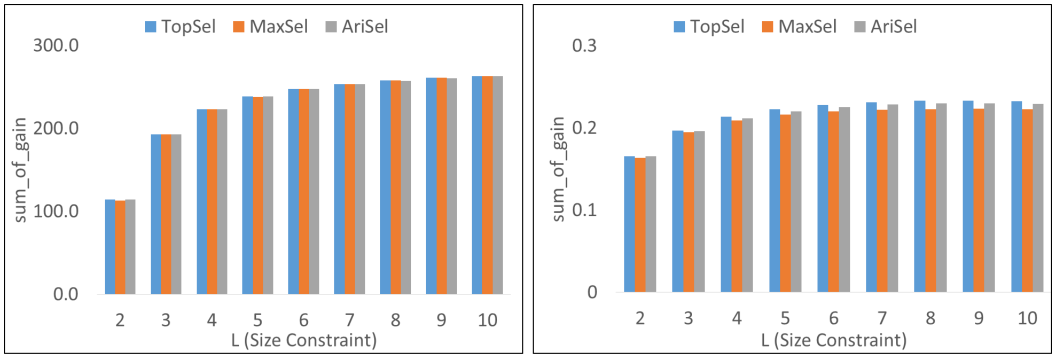


Figure 7.4: Average of profit gain of all candidates in $\mathcal{S}$

profit gain of *AriSel* is up to 0.2% (absolute) less than the *TopSel*. However, the former can return more diverse bundles, indicating by a lower average Jaccard value when $L > 6$ (Figure 7.5a). Having more diverse and equally profitable bundles in $\mathcal{S}$ reasons the profit gain of *AriSel* in Figure 7.3a.

The same observation can be drawn from the relative gain. Although *MaxSel* generates more diverse bundles (Figure 7.5b) than *AriSel*, its lower average profit gain in Figure 7.3b explains why *MaxSel* returns less *sum_of_gain* than *AriSel*. In the subsequent experiments, we use *AriSel* as the generation strategy in our proposed solution.

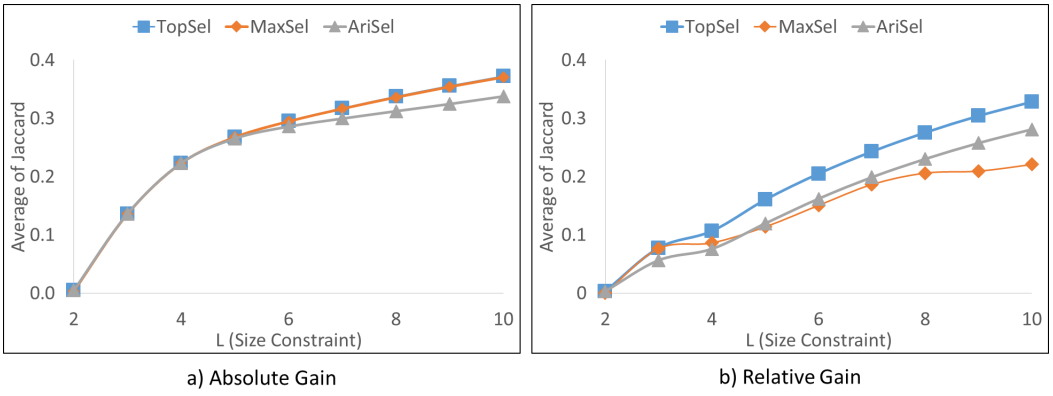


Figure 7.5: Average of Jaccard distances of all candidate pairs

7.4.3 Comparison with Different Cost Factors and Similarity Thresholds

Varying cost factor γ. We examine the effect of different cost factors to the profitability of the final solution. We vary $\gamma \in \{0.1, 0.3, 0.5, 0.7\}$. These γ values reflect a wide range of product cost, in which small γ refers to low cost products, and large γ refers to high cost products. At $\gamma = 0.9$, none of the methods can return bundles with positive gain. Figure 7.6 shows the trend lines of the four γ values above across various size constraints.

It is expected that the sum_of_gain of the final solution decreases when γ increases (Figure 7.6). When γ is low, larger bundles can bring more profit gain. In Figure 7.6a, solutions with more profit gain can be found at $L = 7, \gamma = 0.1$. However, when $\gamma = 0.7$, the profit gain becomes stable at $L = 4$. In terms of the relative gain, the profit gain also converges at $L = 4$ (Figure 7.6b).

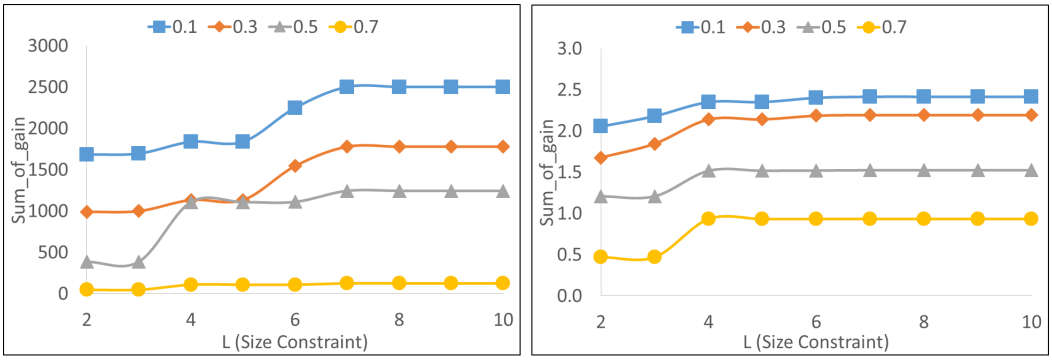


Figure 7.6: Effect of four different cost factor ($\gamma \in \{0.1, 0.3, 0.5, 0.7\}$) on each type of profit gain

Vary similarity threshold θ. We experiment how various $\theta \in \{0.2, 0.4, 0.6\}$ affects the profit gain. The higher the θ is, the more similar bundles we allow in the final solution. Figure 7.7 shows that the more profit gain is returned when θ increases. It again confirms large bundles with high profit gain are highly overlapping. At every $L > 2$, high values of θ return more profit gain (both absolute and relative gain). In Figure 7.7a, when more diverse bundles are preferred ($\theta = 0.2$), increasing L do not bring significant gain in sum_of_gain.

In contrast, when $\theta = 0.4$, increasing L eventually leads to solution with more profit gain. An interesting observation at $\theta = 0.6$ is that large values of L also do not bring more profit gain. It implies that large bundles do not necessarily have more profit gain than small sized bundles.

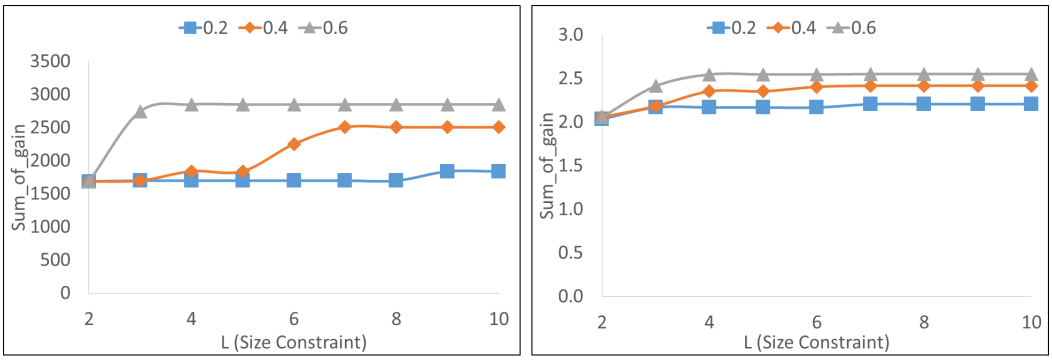


Figure 7.7: Effect of three different similarity thresholds θ ($\theta \in \{0.2, 0.4, 0.6\}$) on each type of profit gain

We verify the high similarity among top profit gain bundles at large sizes ($L > 2$). In Figure 7.8 we plot the average of Jaccard similarity for all bundle pairs in the final solution D. When we enforce high diversity (i.e., $\theta = 0.2$), the top profit gain bundles are mostly non-overlapping with the average of Jaccard closes to zero. When we enforce more similarity (i.e., $\theta = 0.6$), more similar bundles are selected in the final solution. The pairwise Jaccard similarity on average also increases.

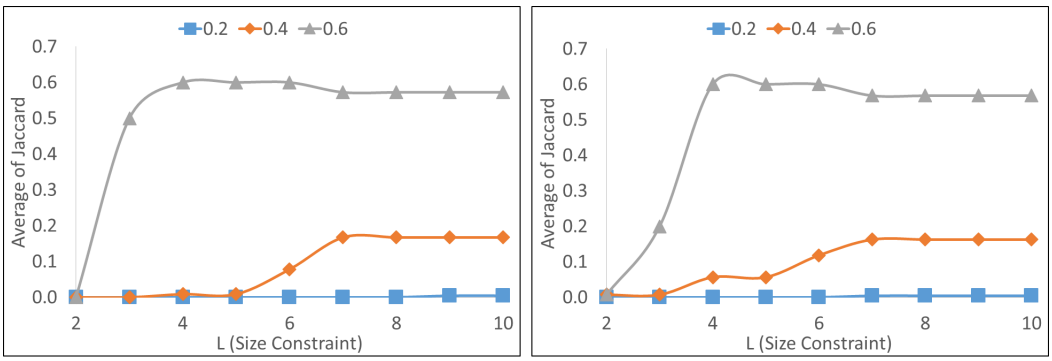


Figure 7.8: Effect of three different similarity thresholds θ ($\theta \in \{0.2, 0.4, 0.6\}$) on the diversity of the top-K results

7.4.4 Comparison against Baseline

Here we compare our solution against the baseline approaches in terms of the total profit gain of the final top-K bundles. Although frequent itemsets and high utility itemsets can be treated as bundles, empirical experiments (same data set, default settings) show that none of the itemsets have positive profit gain. This is as expected since bundles with positive gain are not necessary to be frequent itemsets or high utility itemsets, as explained in Section 7.5.

To compare against the *BundlingConfiguration* algorithm in[27], we create two variants of our solution. The first variant, *TopK-Diverse-Bundles(AriSel, Non-overlapping)*, only return non-overlapping bundles, which can be achieved with a very low θ. With $L = 10$, the smallest possible Jaccard value is $1/20 = 0.05$. Therefore we set $\theta = 0.01$. The second variant, *TopK-Diverse-Bundles(AriSel)*, allows overlapping bundles with $\theta = 0.4$.

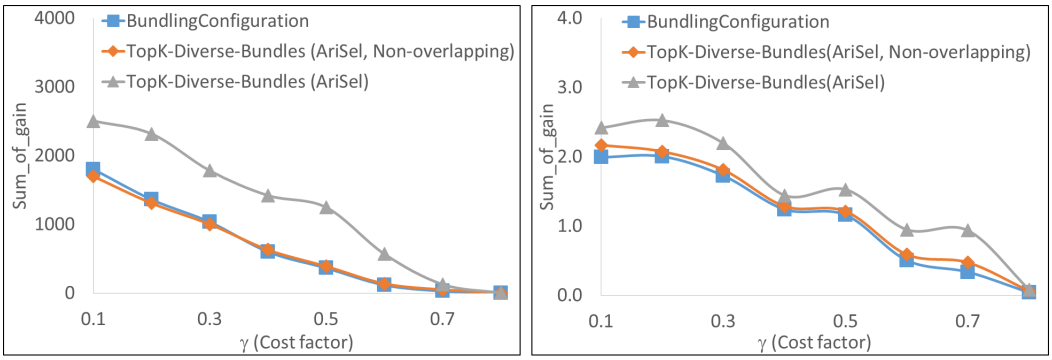
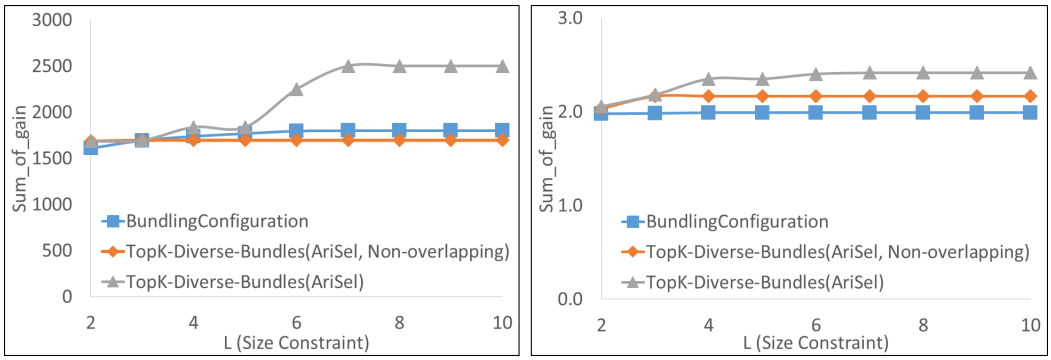


Figure 7.9: Effect of different cost factor γ on profitability of each comparing method

Varying cost factor γ. Figure 7.9 plots the *sum_of_gain* of each of the comparing methods with different cost factor γ. In general, the total profit gain decreases with the increase of cost. *TopK-Diverse-Bundles(AriSel)* always returns more gain solutions than *BundlingConfiguration*. This is expected as the latter is limited to non-overlapping bundles. Even when only non-overlapping bundles are considered, *TopK-Diverse-Bundles(AriSel, Non-overlapping)* does not always return bundles with less gain than the ones

of*BundlingConfiguration*. In Figure 7.9b, our proposed method can find slightly more profit gain than the baseline.

Varying size constraint L. Figure 7.10 plots the sum_of_gain of each of the comparing methods with various size constraint L. Since large bundles are more likely to have more profit gain, only *TopK-Diverse-Bundles(AriSel)* can benefit from overlapping bundles to return higher gains than the other two. Across different Ls, *BundlingConfiguration* returns solution with more absolute gain than *TopK-Diverse-Bundles(AriSel, Non-overlapping)*, but loses to the latter in terms of the relative gain.

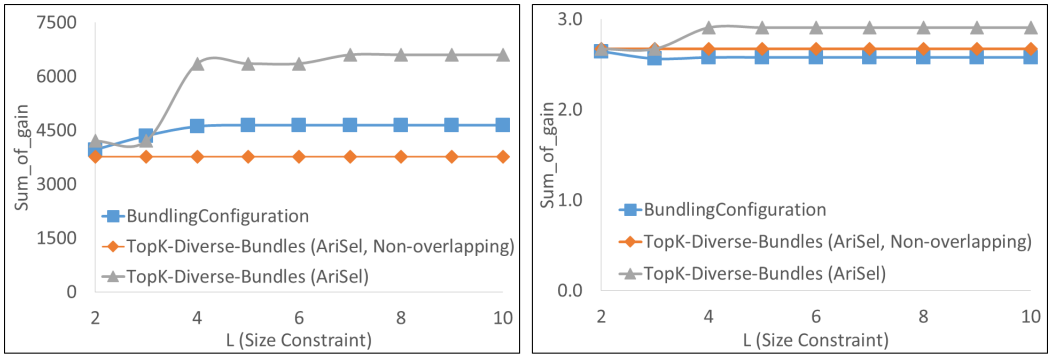


a) Absolute gain b) Relative gain

Figure 7.10: Effect of different size constraints L on profitability of each comparing method

In Table 7.11, we conduct the same experiment set in Table 7.10 on review-based willingness to pay data extracted from Chapter 5. Similar trends can be observed. The *TopK-Diverse-Bundles(AriSel)* still returns higher sum_of_gain than the other two methods. *BundlingConfiguration* has higher absolute gain but lower relative than *TopK-Diverse-Bundles(AriSel, Non-overlapping)*.

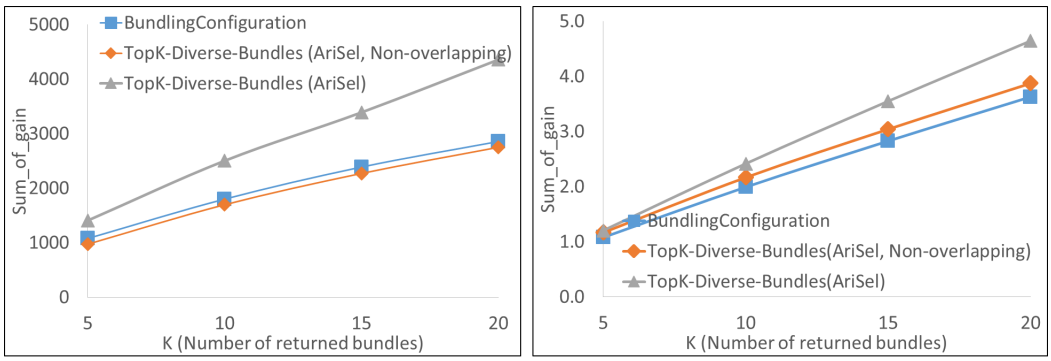
Varying number of returned bundles K. Figure 7.12 plots the sum_of_gain of each of the comparing methods with different number of returned bundles K. *TopK-Diverse-Bundles(AriSel)* returns more bundles with high profit gain than the other two across all K values. The difference in sum_of_gain can be explained by overlapping bundles with high profit gain, which both *Bundling-Configuration* and *TopK-Diverse-Bundles(AriSel, Non-overlapping)* cannot re-



a) Absolute gain b) Relative gain

Figure 7.11: Effect of different size constraints L on profitability of each comparing method (review-based willingness to pay)

turn.



a) Absolute gain b) Relative gain

Figure 7.12: Effect of K on profitability of each comparing method

7.4.5 Case Study

We examine top-10 profit gain bundles of at most 5 components returned by both type of profit gain in Table 7.2 and Table 7.3. For absolute gain, Table 7.2 shows that some of the bundles with high profit gain consist of products in the same category, for example, bundle #10 contains 2 RAM sticks, bundle #5 contains 2 hard drive units. There are also other bundles of products in different categories, such as bundle #7 of a hard drive and a monitor. These observations imply that bundling products within categories, or cross categories can result in more profit. Similar observations are drawn in Table 7.3 with relative gain. Besides, the most profit gain bundles differ between two

tables. A bundle with most absolute gain is not necessary to be a bundle with most relative gain, and vice versa.

In both tables, there are bundles at various size, ranging from 2 to 5. Bundles with more components are not necessary to have more profit gain than bundles with less components. For example, bundle #4 in Table 7.3 is a size-3 bundle. Meanwhile, bundle #6 has 4 components. It motivates considering bundles at different sizes to have solutions with more profit gain.

Table 7.2: Top-10 Most Absolute Gain with Overlapping Bundles (similarity threshold $\theta = 0.4$)

ID	Product Name	Absolute Gain
1	Apple Cinema 30-inch HD Flat-Panel Display M9179LL/A (OLD VERSION) Kinesis Freestyle Solo - Mac USB Keyboard - White (KB-700MW-US) HP LP2065 Monitor, 20 Inch LCD, Silver Bezel, Analog - Digital Interface Hard Drive Caddy for Laptop Dell Latitude D620/D630 NEW	$285.28
2	Apple Cinema 30-inch HD Flat-Panel Display M9179LL/A (OLD VERSION) Kinesis Freestyle Solo - Mac USB Keyboard - White (KB-700MW-US) PalmOne Portable Keyboard for Palm V Series Handhelds	$273.58
3	New Tempered Glass 30” Inch Yamakasi 300 Sparta 2560x1600 S-IPS Monitor with speaker YAMAKASI 300 SPARTA MULTI 30” Monitor LCD S-IPS QHD	$195.50
4	Apple Time Capsule 2TB ME177LL/A [NEWEST VERSION] Transcend Information 1TB StoreJet A3 USB 3.0 (TS1TSJ25A3K)	$176.80
5	LaCie 301359U 4TB 4big Quadra eSATA/FireWire 800/Firewire 400/USB 2.0 RAID Hard Drive Iomega UltraMax 34495 Hard Drive eSATA/FireWire 800/FireWire 400/USB 2.0 1.5TB	$159.30
6	Lilliput 668GL 70NP/H/Y 7” On-camera Field HD Monitor For DSLR with HDMI Ypbpr and Composite Input Dell UltraSharp U2713H 104P6 27-Inch LED-lit Monitor	$156.21
7	ioSafe SoloPRO 2 TB USB 2.0/eSATA Fireproof and Waterproof External Hard Drive with 1 Year Data Recovery Service SH2000GB1YR (Black) QNIX QX2710 LED Evolution ll DP Multi 27” 2560x1440 Samsung PLS (LG IPS) Matt Screen HDMI, Display Port, PC 27-inch Monitor *Thunderbolt Display	$154.61
8	Samsung Touch Of Color T260 25.5-inch LCD Monitor Corsair TWIN2X4096-8500C5D Dominator 4GB 2 X 2GB PC2-8500 1066MHz 240-Pin DDR2 CL5 Dual Channel Desktop Memory	$148.20
9	Corsair XMS3 32GB (4x8GB) DDR3 1600 MHz (PC3 12800) Desktop Memory Lenovo ThinkPad 9 Cell Slice 28++ Add-On Battery for ThinkPad Models T410/T510/W510/T420/T520/W520/T430/T530/W530 (0A36304)	$145.90
10	Corsair Vengeance 32GB (4x8GB) DDR3 1600 MHz (PC3 12800) Desktop Memory Patriot Signature DDR3 16GB (2 x 8GB) 1600MHz CL11 (PC3 12800)	$143.39

Table 7.3: Top-10 Most Relative Gain with Overlapping Bundles (similarity threshold $\theta = 0.4$)

Id	Product Name	Relative Gain
1	Eagle Arion ET-AR502-BK 2.1 Speakers with Subwoofer - 4 Inch Drivers, 40Hz to 20kHz, 30 Watts 1GB RAM Memory Upgrade for the Apple MacBook Pro Series 1.83GHz, 2.0GHz, 2.16GHz — MA092LL/A, MA464LL/A 1GB RAM Memory Upgrade for the Acer Aspire 3690, 3680, 3100, 3620, and 5610 Laptops PalmOne Portable Keyboard for Palm V Series Handhelds	0.26
2	Eagle Arion ET-AR502-BK 2.1 Speakers with Subwoofer - 4 Inch Drivers, 40Hz to 20kHz, 30 Watts 512MB PC133 168 pin SDRAM DIMM Memory RAM for Apple eMac, iMac, PowerMac G4 1GB RAM Memory Upgrade for the Acer Aspire 3690, 3680, 3100, 3620, and 5610 Laptops	0.24
3	Eagle Arion ET-AR502-BK 2.1 Speakers with Subwoofer - 4 Inch Drivers, 40Hz to 20kHz, 30 Watts 512MB PC133 168 pin SDRAM DIMM Memory RAM for Apple eMac, iMac, PowerMac G4 1GB RAM Memory Upgrade for the Apple MacBook Pro Series 1.83GHz, 2.0GHz, 2.16GHz — MA092LL/A, MA464LL/A Hard Drive Caddy for Laptop Dell Latitude D620/D630 NEW	0.24
4	Tenda W150M 150Mbps Mini Wireless AP/Router Zuni Digital Connect ZR301 300 Mbps Wi-Fi Router + 1-Port Switch (ZR301F) Samsung 512MB DDR2 PC2-4200U 533MHz 1Rx8 M378T6553CZ3-CD5	0.24
5	Tenda W150M 150Mbps Mini Wireless AP/Router Zuni Digital Connect ZR301 300 Mbps Wi-Fi Router + 1-Port Switch (ZR301F) Hard Drive Caddy for Laptop Dell Latitude D620/D630 NEW 9” White Tablet PC - Google Android 4.0 Touch Capacitive Screen 1.2ghz 8GB WiFi Twin Camera	0.24
6	Tenda W150M 150Mbps Mini Wireless AP/Router Zuni Digital Connect ZR301 300 Mbps Wi-Fi Router + 1-Port Switch (ZR301F) C2G / Cables to Go 01938 RJ45 8pin Modular TAdapter (White) PalmOne Portable Keyboard for Palm V Series Handhelds	0.23
7	Samsung S24B150BL 23.6-Inch Screen LED-Lit Monitor WD Scorpio Blue 320GB Mobile SATA Hard Drive New Dell Studio 1735 1737 Keyboard TR334 0TR334	0.23
8	Supersonic 7-inch tablet SC-75MID T70 Google Android 7” Tablet (Android 2.3)	0.23
9	Sound Science Corp Qsb 30-Watt USB Stereo Speaker Set (TUS510) Big Blue Studio Wireless Bluetooth Speaker	0.22
10	ASUS VK248H-CSM 24-Inch Full-HD LED-Lit LCD Monitor with Integrated Speakers and Webcam Seagate Momentus 5400.2 - Hard Drive - 60GB - Internal - 2.5IN - Ultra ATA/100 -	0.21

7.5 Related Work

We review relevant work in three directions, namely top-K itemsets, profit maximization and bundling concept in management sciences.

Top-K itemsets. In general, top-K problems concern finding an optimal set of K object according to a certain objective, or satisfying some con-

straints [43]. Objects vary from individual items (e.g., products [79]) to item groups (e.g., itemsets). Since bundle in general is an item group, we relate our problem to this direction.

Related work on mining top-K itemsets has considered a variant of objectives. One direction is to look for itemsets with top frequency in transactional data such as top-K frequent patterns [108, 110]. Instead of frequency, top-K high utility itemset mining [101] returns itemsets with total utility less than at most $K-1$ possible itemsets. Another direction is to recommend top-K most relevant packages to a given consumer preferences [66, 111]. Yet another direction is to find top-K maximal cliques [114].

Our work has a different objective than other top-K itemset problems. We address the top-K profitable bundles. The problem involves a pricing mechanism which has not been considered in the aforementioned works. Hence, other types of top-K itemsets are not necessarily profitable bundles. In particular, frequent itemsets or high utility itemsets are determined based on transactions (which are consumers in our context) containing them. Meanwhile, a profitable bundle may not require at least one consumer to have positive willingness-to-pay on every of its components. Top-K most relevant packages and maximal cliques do not take the production cost into consideration, which is esssential in determining profit.

Diversifying top-K results is an orthogonal research to the top-K problems. The common assumption is that a set of objects with their ranking score is given in advance. To enhance diversity in the top-K results, two main approaches are considered. On the first one, a new ranking score is computed based on diversity and the given ranking score [5, 7]. Top-K results are selected according to the new scores. The second one is a more general framework to handle diversity in top-K results [81]. Given a similarity measure on two objects, and a similarity threshold, the returned list does not contain any pairs which are more similar than the given threshold.

Even though our work follows the proposed framework in the second approach to return diverse bundles, our problem considers a different setting. No bundles and their profitability are given in advance. We first have to compute a list of bundles, then apply the framework to diversify the final results.

Profit maximization objective. Profit maximization has been considered in non-bundling ways in the literature. The key difference between ours and those works is on the computational complexity. We consider a market setting where the seller has multiple units to serve multiple consumers. This distinguishes our work from combinatorial auction [26], in which each item has only one unit, or skyline analysis [107], in which consumers are not taken into evaluating the profit. Although [11, 34] consider a similar market setting to ours, they only focus on determining prices for single items.

Economics, Marketing, and Management Sciences. The effect of bundling on profit has been approached in two different ways in the literature. One line of work examines various factors contributed to bundling profitability, for example, distributions of consumer willingness-to-pay on standalone products (e.g., discrete [1], normal [90]), product cost [82], etc. In these works, analytical models have been devised, mostly for explanatory purposes. On the other hand, another direction investigates different techniques to bundle pricing, such as mixed integer linear programming [36], probabilistic approaches [105]. The techniques were considered in different contexts, for example, a given set of bundles.

Our work can be contrasted in three perspectives. First, our objective is to construct top-K profitable bundles, rather than explanatory purposes. Our problem, as well as the proposed solutions work well with heterogeneity in consumer willingness-to-pay and production cost. Secondly, although bundle pricing is a part of our problem, no bundles is given in advance. Instead, we construct the solution from a given set of standalone products. Thirdly, most of these works consider size-2 bundles, or make simplified assumptions to reduce

computational complexity significantly [9]. In contrast, our only constraint on bundle size is user-specified. Hence, the problem is more computationally challenging when a large number of standalone products is involved.

7.6 Summary

We address the top-K bundles problem by mining willingness-to-pay from rating data. Our objective is to determine the top-K diverse bundles with the most profit gain. We consider two types of profit gain, namely absolute gain and relative gain. We show that the problem is NP-hard even when $K = 1$, and introduce a two-phase solution, in which profitable bundles are first generated in the first phase, and diverse bundles are then selected from the generated bundles. Experiments shows that our approach results in more profitable top-K bundles than the baseline, as well as bundles based on frequent itemsets.

Chapter 8

Conclusion and Future Work

8.1 Summary

The fact that consumers have diverse preferences is important to businesses to improve their products or services. The ever-increasing scale of product review data from e-Commerce websites offers both new research opportunities and challenges to learn about diversity of consumer preferences. How to leverage the diverse preferences from these data effectively and efficiently to improve existing applications still remains an open question. We address the question by developing new models and efficient algorithms to mine diversity of preferences from review data. In particular, our work has two parts: (i) leveraging diversity of preferences between consumers across products to provide better recommendations, and (ii) leveraging diversity of preferences between products across consumers to design profitable products. We summarize the two parts as follows.

Part I includes Chapters 3 and 4. In this part we examine the diversity of preferences between consumers across products from rating data. The existing view of diversity in the literature is to measure the correlation between the co-ratings as an indicator of similarity in preference. We propose to contextualize the diversity. Our intuition is that the diversity may vary in different contexts.

In Chapter 3, we address the signals of diversity of preferences. Since ratings indicate consumer preferences on products, we use difference between the two ratings to signal the diversity of preferences. This approach raises an issue when the rating difference is missing, i.e., either of the ratings is not observed. Hence, we develop *DPMF*, a matrix factorization-based methods to predict the missing rating differences from observed ones. The model requires less parameters, less time to train, and is more effective in predicting rating differences compared to other baselines.

With the rating differences estimated in the Chapter 3, Chapter 4 continues on measuring the similarity of preferences from rating differences. Due to the variance in rating differences between different consumer pairs, we propose a data-driven similarity model, *CAM*, which is learned from each consumer pair's rating difference. Through empirical study, the combination of *CAM-DPMF* provides a better predictive performance than Cosine similarity and Pearson correlation coefficient in an application of user-based Collaborative Filtering, which substantiates the need of contextualizing diversity.

Part II includes Chapters 5, 6 and 7. In this part we examine the diversity of preferences between products across consumers from willingness-to-pay data. We tackle the bundle set design problem, which poses computational challenges due to the number of possible combinations from a given product set.

In Chapter 5, we focus on the problem of estimating willingness-to-pay from review data. Willingness-to-pay data is often proprietary. With the prevalence of review data, we investigate two different approaches. The first approach is based on our intuition that the more a consumer enjoys a product, the more he or she is willing to pay for it. Hence, we propose a linear transformation from ratings to willingness-to-pay. The second approach is to estimate willingness-to-pay from both rating and review data. This approach consists of different modeling techniques from different areas, such as Conjoint Analysis (from Marketing Science), SVM (from Machine Learning), and hier-

archical Latent Dirichlet Allocation (from Text Mining). The distribution of the estimated willingness-to-pay in a category reflects the price range across different products in the same category.

Given willingness-to-pay data, in Chapter 6, we investigate the computational challenge in finding a partition of a given product set to maximize the total profit, and satisfy some bundle constraints. Through a mapping to a graph representation, we prove the problem can be optimally solved when only bundles of size 2 are allowed. For larger bundles, the problem is NP-Hard. Therefore, we propose several heuristic algorithms to efficiently solve the problem. In terms of profitability, we find that allowing large bundles results in more profitable solutions than selling products separately. In terms of scalability, the proposed algorithms scale linearly with the number of consumers, and polynomially with the number of products.

Lastly, in Chapter 7, we investigate another type of bundle set, which is top-K diverse bundles with maximum profit gain. The problem of top-K diverse bundles with the most profit gain is also NP-Hard. We propose a two-phase solution to address the problem efficiently. By comparing against the method in Chapter 6, the proposed solution returns greater profit gain solutions since it is not constrained to non-overlapping bundles.

To summarize, the main contribution in this dissertation is to advocate the benefit of leveraging diversity of preferences to improve existing applications.

8.2 Future Work

We outline below several potential directions for future research that can further improve the current work.

First, in estimating the willingness-to-pay from review data in Chapter 5, we lack of a ground-truth to validate the accuracy of the estimation method. A possible further investigation is to conduct user study in which the consumers

in the study may be asked to write reviews and assign a willingness to pay (to be used as ground truth).

Secondly, we make several assumptions in estimating the consumer's purchase behavior, as well as the monopoly market. Relaxing each of these assumptions would lead to a different purchase behavior, which may not work with the proposed solutions. We envisage future work of efficient algorithms under relaxation of each assumption.

Thirdly, the proposed heuristics in Chapter 6 and 7 do not come with an approximation bound. The approximation quality so far is measured empirically. Achieving some theoretical bounds for these algorithms is useful in quantifying how close the approximation is to the optimal solution.

Last but not least, we only consider review data in our study. Besides review data, there are also other type preference-elicited data such as click through rate data, transactional data, etc. Exploring the concept of diversity of preferences in other type of preference data is also a potential future work.

Bibliography

[1] William James Adams and Janet L Yellen. Commodity bundling and the burden of monopoly. *The Quarterly Journal of Economics*, pages 475–498, 1976.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2011.

[4] Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer, 2012.

[5] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*, pages 5–14. ACM, 2009.

[6] Amr Ahmed, Bhargav Kanagal, Sandeep Pandey, Vanja Josifovski, Lluis Garcia Pueyo, and Jeff Yuan. Latent factor models with additive and hierarchically-smoothed user preferences. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 385–394, 2013.

[7] Albert Angel and Nick Koudas. Efficient diversity-aware search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 781–792. ACM, 2011.

[8] Nikolay Archak, Anindya Ghose, and Panagiotis G Ipeirotis. Deriving the pricing power of product features by mining consumer reviews. *Management Science*, 57(8):1485–1509, 2011.

[9] Yannis Bakos and Erik Brynjolfsson. Bundling information goods: Pricing, profits, and efficiency. *Management Science*, 45(12):1613–1630, 1999.

[10] Yannis Bakos and Erik Brynjolfsson. Bundling and competition on the internet. *Marketing science*, 19(1):63–82, 2000.

[11] Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 29–35. ACM, 2006.

[12] Daniel Berend and Tamir Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

[13] David Besanko and Ronald R Braeutigam. Microeconomics. hoboken, 2005.

[14] Rushi Bhatt, Vineet Chaoji, and Rajesh Parekh. Predicting product adoption in large-scale social networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1039–1048. ACM, 2010.

[15] Li Bian and Henry Holtzman. Online friend recommendation through personality matching and collaborative filtering. *Proc. of UBICOMM*, pages 230–235, 2011.

[16] Christopher M Bishop and Nasser M Nasrabadi. *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[18] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[19] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[20] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 43–52, 1998.

[21] Christoph Breidert, Michael Hahsler, and Thomas Reutterer. A review of methods for measuring willingness-to-pay. *Innovative Marketing*, 2(4):8–32, 2006.

[22] Christoph Breidert, Michael Hahsler, and Lars Schmidt-Thieme. Reservation price estimation by adaptive conjoint analysis. In *Classificationthe Ubiquitous Challenge*, pages 569–576. Springer, 2005.

[23] Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 443–452. IEEE, 2001.

[24] Barun Chandra and Magnús Halldórsson. Greedy local improvement and weighted set packing approximation. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 169–176. Society for Industrial and Applied Mathematics, 1999.

[25] Jaihak Chung and Vithala R Rao. A general choice model for bundles with multiple-category products: Application to market segmentation and optimal pricing for bundles. *Journal of Marketing Research*, 40(2):115–130, 2003.

[26] Sven De Vries and Rakesh V Vohra. Combinatorial auctions: A survey. *INFORMS Journal on computing*, 15(3):284–309, 2003.

[27] Loc Do, Hady W Lauw, and Ke Wang. Mining revenue-maximizing bundling configuration. *Proceedings of the VLDB Endowment*, 8(5):593–604, 2015.

[28] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[29] Theodoros Evgeniou, Constantinos Boussios, and Giorgos Zacharia. Generalized robust conjoint estimation. *Marketing Science*, 24(3):415–429, 2005.

[30] Hui Fang, Yang Baoy, and Jie Zhang. Misleading opinions provided by advisors: Dishonesty or subjectivity. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1983–1989, 2013.

[31] José M Gil and Mercedes Sánchez. Consumer preferences for wine attributes: a conjoint approach. *British Food Journal*, 99(1):3–11, 1997.

[32] Thomas F Golob, Eugene T Canty, Richard L Gustafson, and Joseph E Vitt. An analysis of consumer preferences for a public transportation system. *Transportation Research*, 6(1):81–102, 1972.

[33] Rica Gonen and Daniel Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *EC*, pages 13–20, 2000.

[34] Alexander Grigoriev, Joyce Van Loon, Maxim Sviridenko, Marc Uetz, and Tjark Vredeveld. Bundle pricing with comparable items. In *European Symposium on Algorithms*, pages 475–486. Springer, 2007.

[35] Joseph P Guiltinan. The price bundling of services: A normative framework. *Journal of Marketing*, 51(2), 1987.

[36] Ward Hanson and R Kipp Martin. Optimal bundle pricing. *Management Science*, 36(2):155–174, 1990.

[37] Trevor J. Hastie, Robert John Tibshirani, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2011.

[38] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems*, pages 115–132, 1999.

[39] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the International ACM SIGIR Conference*, pages 259–266, 2003.

[40] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[41] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[42] Hao Huang, Po-Shen Loh, and Benny Sudakov. The size of a hypergraph and its matching number. *Combinatorics, Probability & Computing*, 21(3):442–450, 2012.

[43] Ihab F Ilyas, George Beskales, and Mohamed A Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11, 2008.

[44] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[45] Kamel Jedidi, Sharan Jagpal, and Puneet Manchanda. Measuring heterogeneous reservation prices for product bundles. *Marketing Science*, 22(1):107–130, 2003.

[46] Kamel Jedidi and Z John Zhang. Augmenting conjoint analysis to estimate consumer reservation price. *Management Science*, 48(10):1350–1368, 2002.

[47] Rong Jin, Joyce Y. Chai, and Luo Si. An automatic weighting scheme for collaborative filtering. In *Proceedings of the International ACM SIGIR Conference*, pages 337–344, 2004.

[48] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[49] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[50] Akihisa Kako, Takao Ono, Tomio Hirata, and Magnús M Halldórsson. Approximation algorithms for the weighted independent set problem in sparse graphs. *Discrete Applied Mathematics*, 157(4):617–626, 2009.

[51] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization

for context-aware collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 79–86, 2010.

[52] Anil Kaul and Vithala R Rao. Research for product positioning and design decisions: An integrative review. *International Journal of research in Marketing*, 12(4):293–320, 1995.

[53] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, pages 81–93, 1938.

[54] Uzma Khan and Ravi Dhar. Price-framing effects on the purchase of hedonic and utilitarian bundles. *Journal of Marketing Research*, 47(6):1090–1099, 2010.

[55] Rajeev Kohli and Ramamirtham Sukumar. Heuristics for product-line design using conjoint analysis. *Management Science*, 36(12):1464–1478, 1990.

[56] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[57] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):42–49, 2009.

[58] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3(19):8, 2012.

[59] Kelvin J Lancaster. A new approach to consumer theory. *The journal of political economy*, pages 132–157, 1966.

[60] Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 601–608, 2009.

[61] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[62] Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F Mokbel. Lars: A location-aware recommender system. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 450–461, 2012.

[63] Beibei Li, Anindya Ghose, and Panagiotis G Ipeirotis. Towards a theory model for product search. In *Proceedings of the 20th international conference on World wide web*, pages 327–336. ACM, 2011.

[64] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[65] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[66] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. Personalized travel package recommendation. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 407–416. IEEE, 2011.

[67] Yuping Liu. The long-term impact of loyalty programs on consumer purchase behavior and loyalty. *Journal of Marketing*, 71(4):19–35, 2007.

[68] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[69] Maria Lus Loureiro and Jill J McCluskey. Consumer preferences and willingness to pay for food labeling: A discussion of empirical studies. *Journal of Food Distribution Research*, 34(3):95–102, 2000.

[70] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the International ACM SIGIR Conference*, pages 203–210, 2009.

[71] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 287–296, 2011.

[72] Lester W Mackey, David Weiss, and Michael I Jordan. Mixed membership matrix factorization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 711–718, 2010.

[73] N Mankiw. *Principles of economics*. Cengage Learning, Stamford, CT, 2015.

[74] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2015.

[75] Rokia Missaoui, Petko Valtchev, Chabane Djeraba, and Mehdi Adda. Toward recommendation based on ontology-powered web-usage mining. *IEEE Internet Computing*, 11(4):45–52, 2007.

[76] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2007.

[77] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 37–76. Springer, 2015.

[78] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web*, pages 325–341. Springer, 2007.

[79] Yu Peng, Raymond Chi-Wing Wong, and Qian Wan. Finding top-k preferable products. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1774–1788, 2012.

[80] Jim Pitman et al. Combinatorial stochastic processes. 2002.

[81] Lu Qin, Jeffrey Xu Yu, and Lijun Chang. Diversifying top-k results. *Proceedings of the VLDB Endowment*, 5(11):1124–1135, 2012.

[82] Vithala R Rao. *Handbook of pricing research in marketing.* Edward Elgar Publishing, 2009.

[83] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 81–90, 2010.

[84] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW)*, pages 175–186, 1994.

[85] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook.* Springer, 2015.

[86] Jennifer Roback. Wages, rents, and the quality of life. *The Journal of Political Economy*, pages 1257–1278, 1982.

[87] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 880–887, 2008.

[88] Dominick Salvatore et al. Microeconomics: theory and applications. *OUP Catalogue*, 2008.

[89] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings*

of the ACM International Conference on World Wide Web (WWW), pages 285–295, 2001.

[90] Richard Schmalensee. Gaussian demand and commodity bundling. *Journal of Business*, pages S211–S230, 1984.

[91] Hanhuai Shan, Jens Kattge, Peter B Reich, Arindam Banerjee, Franziska Schrodt, and Markus Reichstein. Gap filling in the plant kingdom trait prediction using hierarchical probabilistic matrix factorization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1303–1310, 2012.

[92] Börkur Sigurbjörnsson and Roelof Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pages 327–336. ACM, 2008.

[93] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[94] Ping-Han Soh, Yu-Chieh Lin, and Ming-Syan Chen. Recommendation for online social feeds by exploiting user response behavior. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 197–198. ACM, 2013.

[95] Michael R Solomon. *Consumer behavior: Buying, having, and being*. prentice Hall Engelwood Cliffs, NJ, 2014.

[96] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 1329–1336, 2004.

[97] George J Stigler. United states v. loew's inc.: A note on block-booking. *Sup. Ct. Rev.*, page 152, 1963.

[98] Jiliang Tang, Xia Hu, and Huan Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013.

[99] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 2012.

[100] Vincent S Tseng, Bai-En Shie, Cheng-Wei Wu, and S Yu Philip. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE transactions on knowledge and data engineering*, 25(8):1772–1786, 2013.

[101] Vincent S Tseng, Cheng-Wei Wu, Philippe Fournier-Viger, and S Yu Philip. Efficient algorithms for mining top-k high utility itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):54–67, 2016.

[102] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[103] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[104] R Venkatesh and Wagner A Kamakura. Optimal bundling and pricing under a monopoly: Contrasting complements and substitutes from independently valued products. 2003.

[105] R Venkatesh and Vijay Mahajan. A probabilistic approach to pricing a bundle of products or services. *Journal of Marketing Research*, pages 494–508, 1993.

[106] R Venkatesh and Vijay Mahajan. 11 the design and pricing of bundles: a review of normative guidelines and practical approaches. *Handbook of Pricing Research in Marketing*, page 232, 2009.

[107] Qian Wan, Raymond Chi-Wing Wong, and Yu Peng. Finding top-k profitable products. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1055–1066. IEEE, 2011.

[108] Jianyong Wang, Jiawei Han, Ying Lu, and Petre Tzvetkov. Tfp: An efficient algorithm for mining top-k frequent closed itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):652–663, 2005.

[109] Cathy R Wessells, Robert J Johnston, and Holger Donath. Assessing consumer preferences for ecolabeled seafood: the influence of species, certifier, and household attributes. *American Journal of Agricultural Economics*, 81(5):1084–1089, 1999.

[110] Raymond Chi-Wing Wong and Ada Wai-Chee Fu. Mining top-k frequent itemsets from data streams. *Data Mining and Knowledge Discovery*, 13(2):193–217, 2006.

[111] Min Xie, Laks VS Lakshmanan, and Peter T Wood. Generating top-k packages via preference elicitation. *Proceedings of the VLDB Endowment*, 7(14):1941–1952, 2014.

[112] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, volume 10, pages 211–222. SIAM, 2010.

[113] Bo Yang, Tao Mei, Xian-Sheng Hua, Linjun Yang, Shi-Qiang Yang, and Mingjing Li. Online video recommendation based on multimodal fusion and relevance feedback. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 73–80. ACM, 2007.

[114] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Finding top-k maximal cliques in an uncertain graph. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 649–652. IEEE, 2010.